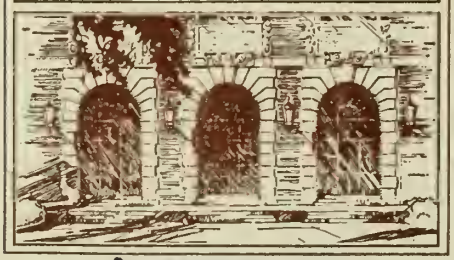


UIUCDCS-R-76-825

AN INVESTIGATION OF THE APPLICATION OF MICROPROCESSORS TO LOW COST AREA NAVIGATION CAPABILITIES FOR GENERAL AVAIATION

BY

Wayne Douglass Smith

September 1976

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

Report No. UIUCDCS-R-76-825

AN INVESTIGATION OF THE APPLICATION OF MICROPROCESSORS TO LOW COST AREA NAVIGATION CAPABILITIES FOR GENERAL AVIATION

by

Wayne Douglass Smith

September 1976

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

This work was submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science, October 1976.

ACKNOWLEDGMENTS

The author would like to acknowledge the help of those individuals who contributed so much of their time and energy to assist in the preparation of this thesis. I would first like to thank all the members of my thesis committee who provided the guidance and assistance which made the work possible. Professor Thomas A. Murrell receives special thanks for his patience, understanding, and active participation throughout the entire project. Without his insight and direction, the problems would have been insurmountable. Professor Michael Faiman also deserves particular thanks for his invaluable assistance with the logic design phase of the project.

The author would also like to express his gratitude to the many other individuals who assisted with the project. Professor Richard Montanelli provided much needed assistance in interpreting the statistical data from the project. Mr. John Bail of Narco Avionics assisted by providing several technical manuals, and a great deal of encouragement.

I would like to thank my wife, Anne, who typed and helped prepare the original drafts of the thesis. Mrs. Gayanne Carpenter receives thanks, not only for typing the final copy of the thesis, but for providing encouragement when it was most needed. I must also thank my daughters, Leigh Anne and Erin, for their many hours of reading hexadecimal core dumps for their father to check.

Finally, I would like to acknowledge the assistance of my father, Mr. Frank E. Smith, who provided much of the inspiration for this undertaking, but did not see its completion.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

1. INTRODUCTION

1.1. General Introduction

The research presented in this thesis concerns a study of the application of microprocessors to provide improved aerial navigation capabilities for general aviation. For the purposes of this thesis, general aviation is defined as that segment of avaiation not directly related to commercial or military opeartions. In 1970, there were approximately 130,000 general aviation aircraft in the United States. The majority of these (83 percent) were single-engined aircraft [1].

As air traffic has increased throughout the years, successively more complex air route structures and air traffic control procedures have been imposed by the Federal Aviation Administration. These structures and procedures are essential for air safety, and to insure the equitable allocation of airspace resources to all users. With these more stringent operational requirements, however, has come the need for increased accuracy in air navigation techniques. This increased accuracy has been brought about primarily through the use of ground based radio aids to navigation.

The VORTAC radio navigation system, currently in use in the United States, is rapidly approaching obsolescence. A joint FAA/Industry task force has recommended that a new system, Area Navigation, be installed in all aircraft by 1982 [2]. This system would utilize the existing ground based facilities, and would provide significant improvements in the navigational capabilities of the aircraft so equipped.

This Area Navigation System has already gained wide acceptance in commercial and military aviation. This equipment is, however, quite expensive. Area Navigation Systems range in price from about $2,000 to $150,000, with the capabilities of the equipment directly related to the price[2].

Due at least in part to the high cost of these systems, Area Navigation has not yet received wide acceptance in general aviation. The purpose of this thesis is to show that an Area Navigation System based on an off-the-shelf microprocessor can offer a low cost alternative to current special purpose systems.

This thesis is primarily concerned with the application of microprocessors to the solution of a specific problem, that of Area Navigation computations. In such an application, however, many problems must be solved that are of a much more general nature. The problems of slow speed, short word lengths, and limited arithmetic capabilities are common to almost all microprocessor applications. The specific design of the R-Nav System will be covered in detail. At the same time, some general conclusions will be made that are applicable to the wider question of using microprocessors for other applications.

1.2. Current Radio Navigation Procedures

The VORTAC system of radio navigation has been adopted as the standard short range navigation system, both internationally, and in the United States. This system, which is explained in detail in Chapter 2, provides both bearing and distance information to the crew. Both signals are in relation to the ground station to which the airborne radios are tuned.

The bearing information is the angle measured from magnetic North, through the station, to the aircraft position. The distance figure is the straight line distance between the aircraft and the station. Since the aircraft is normally higher than the station, this value does not represent the true norizontal distance to the station, but rather a slant-range distance.

Since the VORTAC stations operate in the VHP and higher bands, their range is limited to line of sight. Many of these ground stations are required to support navigation over a large area. There are approximately 700 such stations in the United States [3]. In order to provide facilities for long range navigation, a system of airways has been established that connects these stations in a network that covers most of the United States. All aircraft operating above 18,000 feet or within certain restricted airspace are required to utilize this airway structure.

1.3. Area Navigation Systems

An Area Navigation System utilizes the information provided by the current VORTAC system to provide greatly expanded navigational capabilities. The bearing and distance information provided by the VORTAC station can be thought of as a vector in a polar coordinate system in which the location of the ground station is the origin. This RHO/THETA system allows the pilot to fix his position exactly (ignoring, or correcting for, the slant-range error in the distance figure).

Any point within the range of the VORTAC station can also be expressed in this RHO/THETA system. With both points specified in this manner, the computation of the bearing and distance from the aircraft's current position to any other point becomes a matter of solving for the unknowns in an oblique triangle. Two sides and the included angle are known. The distance to the desired point, which is called a waypoint, may be computed by using the law of COSINES. The bearing (angle) can then be computed using the law of SINES [4].

When these calculations are performed rapidly and accurately, the aircraft is no longer restricted to navigating directly to or from the VORTAC station. Any point within the range of the station then becomes a usable navigational fix. Both analog and digital computers have been adapted for on-board use to perform these calculations. These computers are interfaced directly to the VORTAC radios for input, and to the aircraft's navigational instruments for output.

Such a computer-based navigation system is called an Area Navigation System. Since an aircraft using this system can navigate to any point within the range of the station, they are also called Random (point) Navigation, or R-Nav Systems. Current trends in R-Nav Systems are towards digital, or in some cases, hybrid computers. Most current R-Nav Systems utilize special purpose computers, designed specifically for R-Nav applications.

1.4. Problems With the Current Airway System

The air route structure previously described has several shortcomings. Organized as it is, on a station-to-station basis, aircraft going more than a few hundred miles can seldom navigate in the most direct (great circle) route from departure to destination. Instead, the aircraft must proceed in a zigzag pattern from station to station. This procedure results in longer flight times than would be necessary with direct routes.

The airway structure also forces unnecessary congestion directly over the VORTAC stations. Due to the arrangement of the airway system, airway intersections are normally located over the stations. This forces aircraft on widely divergent flight paths to navigate directly through the same airspace.

Since the airway system was originally designed for use by aircraft which were not equipped with the distance measuring capability of VORTAC, the system does not take full advantage of this feature. The Distance Measuring Equipment is seldom used other than as holding point, a reporting point, or an approach fix.

In general, the airway system represents an inefficient method for utilizing the available airspace. The airways themselves are overcrowded, while large areas nearby are completely unused. Added to these problems is the "cone of confusion" immediately over the stations themselves.

This is an area immediately above the station, where bearing signals are unreliable. This cone may be as much as 100 degrees in width [5]. The airway system forces air traffic directly through these cones. This means that navigation information is unreliable at the point of greatest traffic concentration, the point where reliable navigation is most important.

1.5. Area Navigation as a Solution to the Airway Problems

R-Nav Systems have the necessary capabilities to alleviate all the problems outlined above. Aircraft using R-Nav can navigate directly from departure to destination, via the most direct route. This not only decreases flight time, but also reduces congestion, while taking full advantage of available airspace. Faster aircraft will be able to pass slower aircraft by simply flying to a slightly offset waypoint. The cone of confusion can be avoided completely. Finally, each airport can have an almost unlimited number of approach and departure fixes.

As mentioned earlier, a joint FAA/Industry R-Nav Task Force has recommended that R-Nav Systems become mandatory for aircraft operating in the high altitude (above 17,000 feet) structure by 1977. The Task Force further recommended that R-Nav be required on all aircraft, at 17,000 feet and below, operating between medium density terminals, by 1982 [3].

With all these attributes, R-Nav must still overcome the price obstacle before gaining wide acceptance with the general aviation population. A review of current R-Nav prices gives the following values for generally available systems [6].

Lowest price $1,995
Average price $13,160
Highest price $51,000.

These prices do not necessarily include the price of the VOR or DME radios, or the display units used to present the navigational information to the crew. The average price of a VOR receiver is $4,775, while that of a DME radio is $11,766. These figures represent the average price of all models of this type equipment listed in a recent publication devoted to this subject [6].

From these figures, it is obvious that the price of an R-Nav System represents a substantial investment, and a considerable percentage of the total aircraft cost. Since R-Nav is not currently required by FAA, there is little wonder that few such systems have found their way into general aviation. The microprocessor-based integrated navigation system outlined in this thesis is one approach to removing the price obstacle and making R-Nav Systems more readily available to general aviation.

2. DETAILED PROBLEM DEFINITION

2.1. Principles of Operation of Very High Frequency Omnidirectional Range (VOR)

VOR receivers are used by airborne aircraft to receive bearing (directional) information transmitted by ground stations. In addition to navigation information, VOR stations also transmit audio signals for identification, weather, and other services.

The basic bearing information transmitted by the VOR station is contained in two 30 Hz signals. One, called the reference signal, is transmitted continuously by an omnidirectional antenna. The other, phase-varying, signal is transmitted by a directional antenna, which is rotated (electronically) at the rate of 30 cycles per second. The aircraft receiver sees the phase-varying signal as time dependent. The amplitude of this signal will assume its maximum value when the rotating antenna is oriented on an azimuth directly from the station to the aircraft. Similarly, the amplitude will be at a minimum when the antenna is on the reciprical of this azimuth. Since the antenna is rotated at the rate of 30 cycles per second, the phase-varying signal is received at the aircraft as a 30 Hz signal.

The phase of the reference signal is adjusted so as to be at a maximum exactly when the rotating antenna passes magnetic North. The two signals, as received at the aircraft, will be out of phase by an angle which is directly proportional to the angular displacement of the aircraft from the station, measured from magnetic North, through the station

(Figure 1). By detecting this phase difference, the VOR receiver can furnish bearing information to the pilot. The detection of this phase difference is currently accomplished, using analog techniques, in the VOR receiver itself [7].

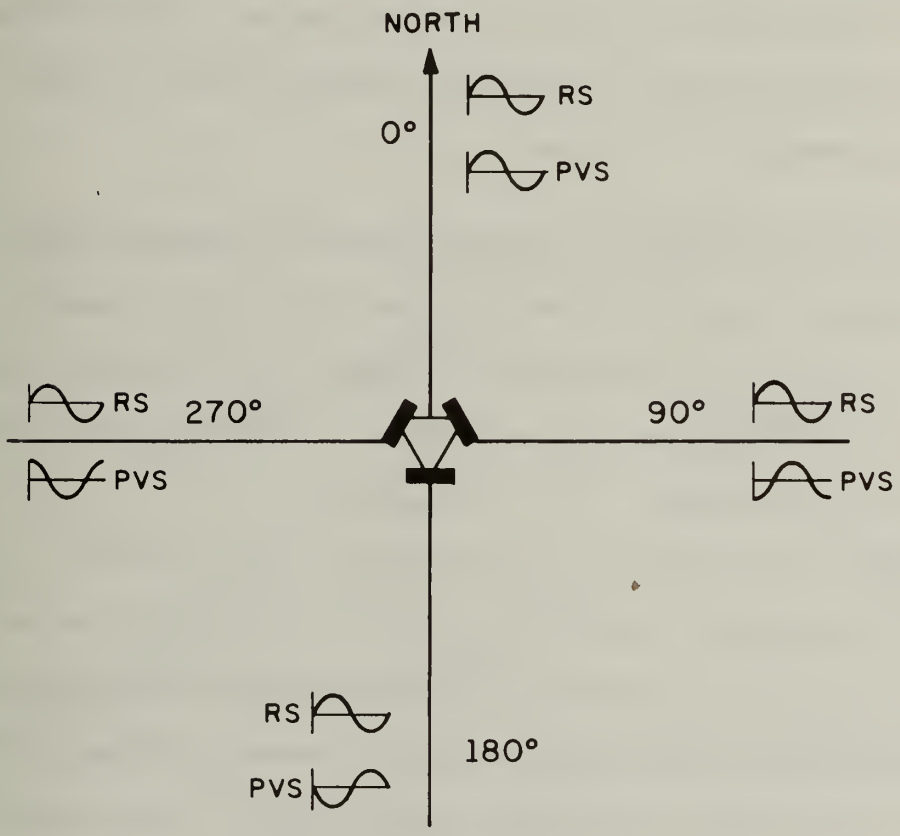


Figure 1. Relationship between VOR signals and bearing from the station.

Bearing information may be presented to the pilot in several ways. The most common form of presentation is through a Course Deviation Indicator (CDI). In the CDI, a vertical needle indicates the angular deviation from a course selected by the pilot. Since a given course exists

on both sides of the station, an ambiguity (TO/FROM) indicator is incorporated into the CDI to indicate whether the selected course will take the aircraft to or from the station. More elaborately equipped aircraft may also contain a Horizontal Situation Indicator (HSI). This instrument has a single needle, much like a compass, which indicates the magnetic bearing from the aircraft to the station. Both CDI's and HSI's may be integrated into the aircraft compass system to provide additional information to the pilot.

2.2. Principles of Operation of Distance Measuring Equipment (DME)

Distance Measuring Equipment is basically a pulse ranging system that operates in the 960-1215 MHz range. In the VORTAC system, the tuning of the DME radios is accomplished in conjunction with the runing of the VOR receiver.

Both the aircraft and the DME ground station operate a transmitter/receiver pair, separated in frequency by 63 MHz. At predetermined intervals, the airborne transmitter sends a pulse which is received by the ground station. The receipt of this "interrogation" pulse triggers the ground transmitter to reply with a similar signal. The airborne equipment measures the time delay between transmission and reception, minus a known 50 μsec delay by the ground station. This time is directly proportional to the distance from the station (Figure 2). Since the aircraft is at a higher altitude than the station, this distance represents

the "slant range" distance to the station, and is measured in nautical miles (6,076 feet). In order to reduce the effects of noise, both transmitters actually transmit a pair of 3.5 μsec pulses, separated by 12 μsec. The receivers are designed to respond only to the correct pulse pairs [7].

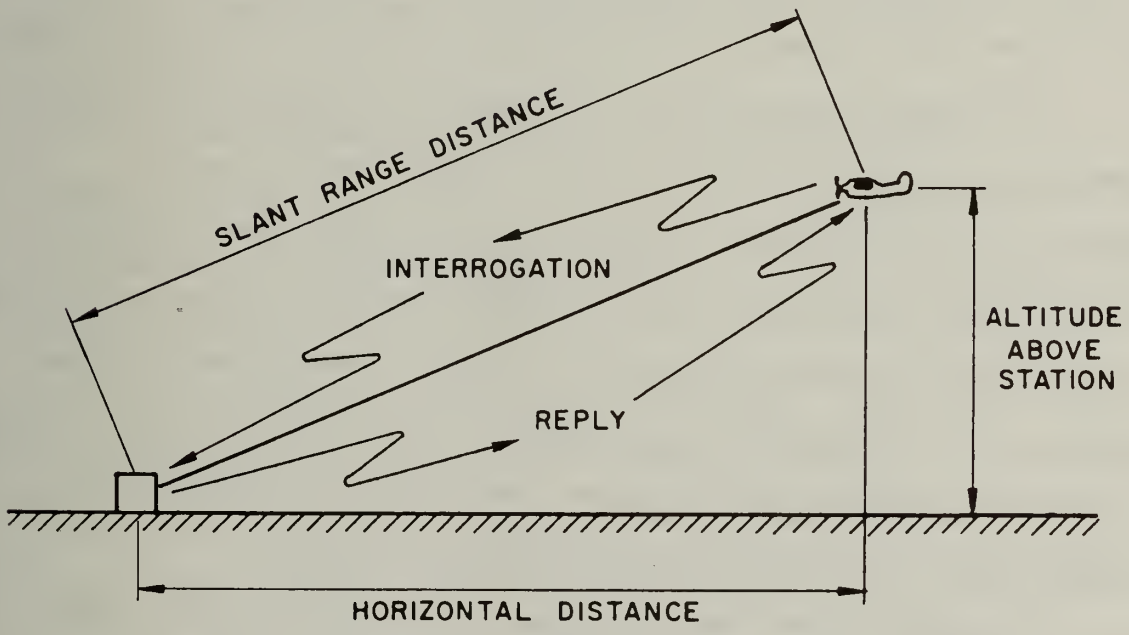


Figure 2. The DME pulse ranging system.

This relatively simple process is complicated considerably by the fact that up to 100 different aircraft may be interrogating the station simultaneously. This means that the airborne equipment receives responses intended for several other aircraft, as well as those from its own interrogations. Added to this problem is the 100 μsec recovery time of the ground transmitter. Should an interrogration from one aircraft fall within 100 μsec after the interrogration from another aircraft, the second interrogation will be ignored. The airborne equipment must, therefore, distinguish between responses to its own interrogations, which do not return in every case, and those intended for up to 99 other aircraft.

This discrimination is made possible by designing the airborne transmitter so as to interrogate at random rates. That is, no two transmitters will interrogate at exactly the same rate. This technique greatly reduces the probability that two pulses intended for another aircraft will occur after the same time interval following transmission by a given aircraft. On the other hand, the probability that the aircraft will receive two of its own responses after the same time interval is reasonably high. The system operation depends on detecting this difference.

The airborne DME system operates in two distinct modes. When the radios are first turned on, or when the station frequency is changed, the system goes into the search mode. This mode is used when there is no previous distance information available to the system. It is used to search the entire range of possible distances in an attempt to determine the current distance from the station. After the distance is established, the track mode is used to check the latest interrogation returns against the last known distance in order to detect small changes of distance.

In the search mode, the DME operates in the following manner. The aircraft transmitter sends pulses at the rate of approximately 150 pulse-pairs per second. The receiver then examines 20 μsec time windows, starting at the time representing 0 miles, and moving the window outward at the rate of 10 miles (120 μsec) per second. The ground transmitter is designed to transmit a maximum of 2,700 pulse-pairs per second. As the window moves, the receiver will receive about 8 random pulses per second. As the window passes through the time area corresponding to the aircraft's own responses, the receiver will pick up almost 30 pulses per second [7].

Once the 30 pulse per second return rate has been received, the DME goes into the track mode. In this mode, the transmitter pulse rate is reduced. As mentioned previously, the pulse rate is variable, but is generally between 2 and 25 pulse-pairs per second. The same 20 μsec time window is retained and examined for returns. If the return falls in the early part of the window, the window is advanced. If the return falls in the late part of the window, the window is delayed. If the pulse is not received at all, the system remains at the last position for up to 10 seconds, awaiting a correct response. If no response is obtained after 10 seconds, the systems returns to the search mode.

Distance information is displayed to the pilot in several ways. One system uses a single-pointer, voltmeter type instrument, usually graduated in two ranges, one of which is selected by the pilot. A more complex system uses a three digit drum instrument, similar to an automobile odometer. Lately, there has been a trend toward seven-segment displays.

2.3. Detailed Analysis of R-Nav Geometry

2.3.1. Introductory Remarks

Aerial navigation uses a slightly different set of references than those customarily used in mathematics. In navigation, the X axis is oriented to the North, and the Y axis to the East. Angles are measured clockwise from the X or North axis. Angles are measuried in degrees between 1 and 360. Since 360 degrees are equivalent to zero degrees, sometimes a system of 0 to 359 degrees is used. While angles greater than 360 or less than 0 are not used in navigation, such angles may occur during mathematical calculations. During the discussion which follows, the assumption will be made that all angles are between 0 and 359.9 degrees, rounded to the nearest tenth. Immediately following any calculation which could produce an angle outside this range, there is an automatic adjustment to bring the angle back within range. This adjustment would consist of adding 360.0 to any negative angle, or subtracting 360.0 from any angle greater than 359.9 degrees. These tests and adjustments are implemented in the system software where required.

Two distinct reference points are used in measuring angles for navigation. In some cases, as when the pilot measures an angle from a map, the angle is in reference to true North. Aircraft compass systems, and consequently most navigation instruments, are referenced to magnetic North. The difference between the two is called variation, and will vary depending on the geographic location in question.

Throughout this thesis, all references will be to magnetic North. This basis was chosen because the VOR bearing used for airborne computation is a magnetic bearing. It will be assumed that when the pilot enters the bearing to a waypoint, this will be adjusted to represent a magnetic bearing. This does not impose undue restrictions on the pilot since all maps and charts are marked with the variation in one degree increments. Since the FAA currently publishes radio navigation information in reference to magnetic North, it is reasonable to assume that when waypoint information is published on FAA charts, it will also be in reference to magnetic North.

The term reciprocal, as used in navigation, also requires some explanation. Unlike the mathematical definition, the reciprocal of a bearing, heading, or course in navigation refers to the same vector, but oriented in the opposite direction. The reciprocal of any bearing is readily obtained by adding or subtracting 180 degrees, whichever produces a positive value between 0 and 359.9 degrees. In the discussion which follows, the pseudo-operation RECIP(X) is assumed to produce the reciprocal of an angle.

Finally, in any navigation system, consideration must be given to the fact that the earth is a sphere rather than a flat surface. For long-range navigation systems, the difference is significant. For short-range navigation systems, such as those discussed in this thesis, the differences are quite small. Since VORTAC stations are only guaranteed frequency protection for 100 nautical miles, the maximum distance involved will not exceed 200 miles. Figure 3 depicts the difference between an

assumed flat earth distance of 200 nautical miles and the actual arc distance, assuming that the earth is a perfect sphere, with a radius of 8,000 nautical miles. This difference is [8]:

$$D = S - C$$

where:

$$S = R \cdot TH$$

with TH in radians. The difference is, therefore:

$$D = 8000 \cdot 2 \cdot SIN^{-1} (100/8000) - 200$$

or about 0.0052087 nautical miles. This distance is much too small to be resolved by any of the navigational instruments under discussion and may be ignored. The remainder of this thesis will be predicated on the assumption of a flat earth.

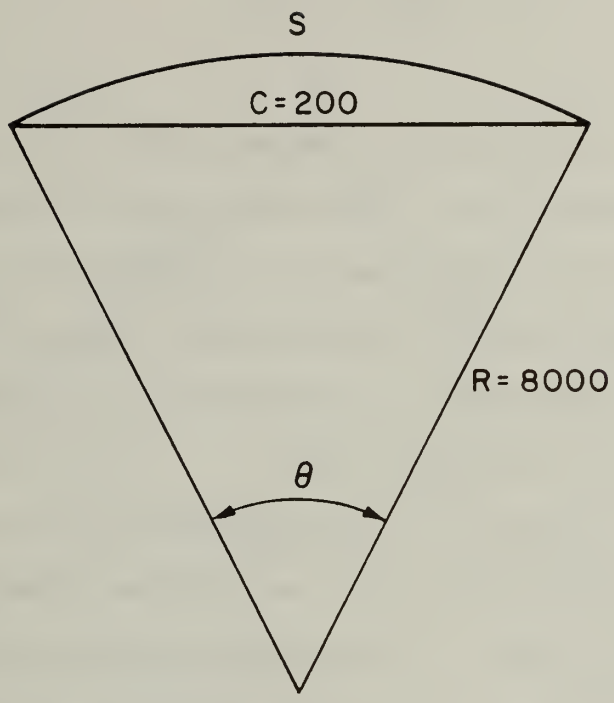


Figure 3. The error in the flat earth assumption

2.3.2. R-Nav Geometry

The initial step in the R-Nav computation is to remove the slant-range error inherent in the DME distance value. This error must be removed, unless the aircraft altitude is very low, in order to produce accurate results. The distance to the waypoint (DW) entered by the pilot will be the true horizontal distance to that waypoint. Therefore, the distance from the station to the aircraft must also be converted to a true horizontal distance. If the altitude of the aircraft above the station is known, then the horizontal distance may be computed using the Pythagorean theorem:

$$HD = \sqrt{SRD^2 - ALT^2}$$

The altitude must, of course, be in the same units as the slant-range distance, i.e. nautical miles. The altitude above the station is easily obtained by the pilot by subtracting the station elevation, which is published on navigation charts, from his flight altitude. One of the tasks which the microrpocessor will perform will be the conversion of the altitude from feet to nautical miles.

The altitude entered by the pilot need only be accurate to the nearest 100 feet, since in the worst case situation (directly over the station), an altitude error of 50 feet represents only a .008 nautical mile error. Errors become successively smaller as the distance from the station increases. Since the proposed navigation system will operate with a resolution of 0.1 mile, a 607 foot error would be required to cause a noticable error. This figure becomes 304 feet if rounding is considered in the computations. Both these tolerances become correspondingly larger as the distance from the station increases.

An error of even this magnitude would not be particularly troublesome except for the fact that a square root computation is involved. Such an error could result in an attempt to compute the square root of a negative number if the altitude is in error on the positive side. This will have to be anticipated in the system software.

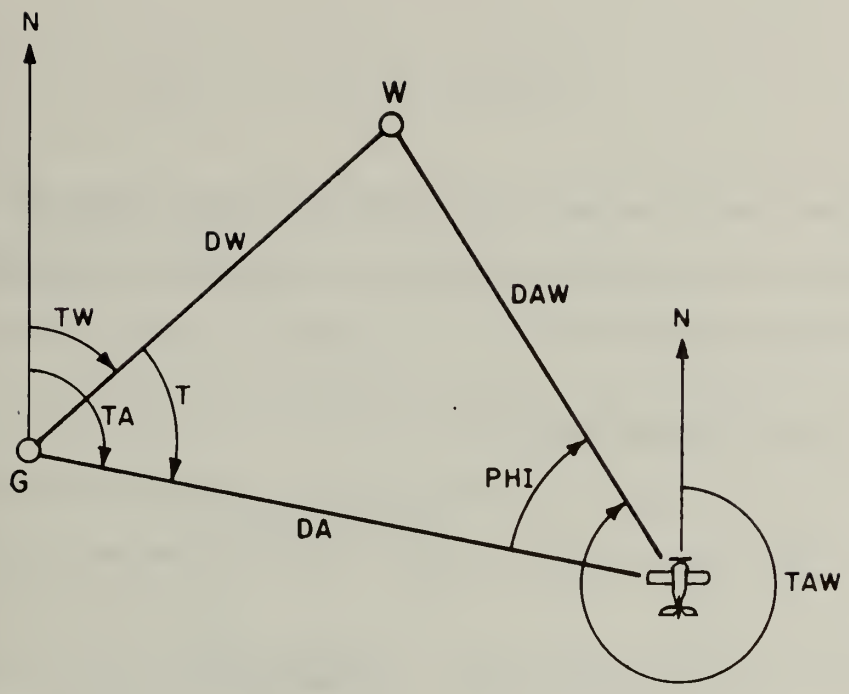


Figure 4. R-Nav Geometry

The geometry of the R-Nav computation is shown in Figure 4. TW is the bearing, and DW is the distance, of the Waypoint, W, relative to the ground station, G. TA and DA are the bearing and distance, respectively, of the aircraft relative to the ground station. The angle TW and the Distance DW are entered by the pilot either prior to take-off or during flight. The angle TA is obtained from the VOR receiver. The Distance DA is the DME Distance from the station corrected for slant-range error.

From this information, the angle T can be computed with the formula:

$$T = TA - TW.$$

At this point, two sides and the included angle are known, and the triangle can be solved. There are two approaches to solving the triangle, each of which is detailed below.

2.3.3. Obtuse Triangle Solution

The first approach to the R-Nav solution was outlined in Chapter 1. The unknown side DAW can be computed from the law of COSINES:

$$DAW = \sqrt{DA^2 + DW^2 - 2 \cdot DA \cdot DW \cdot COS(T)}.$$

Once DAW is known, the angle PHI can be computed, using the law of SINES:

$$PHI = SIN^{-1} (DW \cdot SIN (T) / DAW).$$

A certain amount of care must be exercised at this point, since the above equation has an unlimited number of roots, as long as the argument is less than or equal to 1. For convenience, these can be divided into two groups indicated by the relation:

$$SIN(Y) = SIN(Y+N \cdot 360)$$

and

$$SIN(Y) = SIN(180 - Y + N \cdot 360)$$

where N is an integer, positive, negative, or zero, reflecting the fact that the SIN function is cyclic. Since the angles of concern have already been restricted to a range of 0 through 359.9, N can be assumed to be zero in both cases. The second equation then reduces to:

$$SIN(Y) = SIN(180 - Y)$$

which cannot be so easily dismissed. The situation is illustrated in Figure 5 where two roots are shown as PHI and PHI' and:

$$PHI = 180 - PHI'.$$

This ambiguity cannot be ignored but can be resolved.

First, the remaining angle in the triangle is computed:

$$TOPP = 180 - (T + PHI).$$

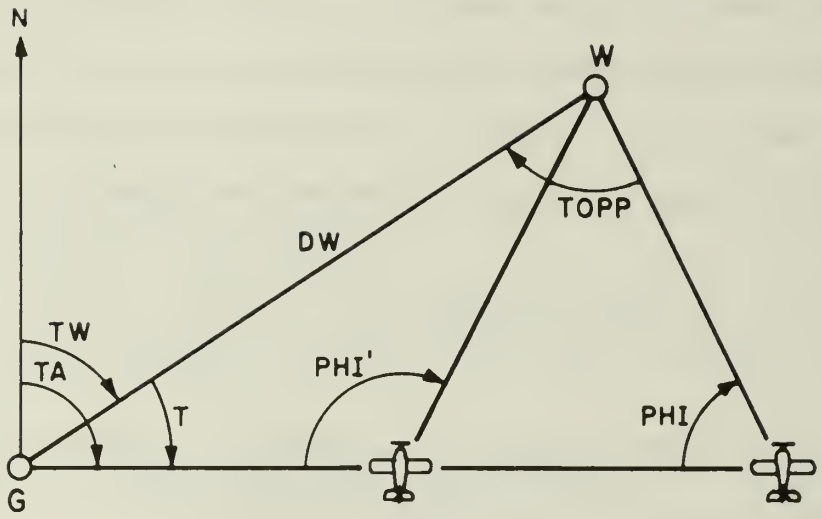


Figure 5. Ambiguity in R-Nav solution.

Using this angle, and the law of COSINES again, a value for DA can be computed.

$$DA = \sqrt{DW^2 + DAW^2 - 2 \cdot DAW \cdot DW \cdot COS(TOPP)}.$$

Since the correct value of DA is known, the computed value can be compared to the known value. If these values do not agree within some reasonable tolerance, then the incorrect value for PHI was used. PHI is then replaced by its supplement, 180 - PHI. This poses some additional computational complexity, but will always resolve the ambiguity.

For the computed information to be of value to the pilot, some additional calculations are necessary. The angle of interest to the pilot is the angle TAW, which is in reference to magnetic North. By observation (Figure 4), this is easily calculated as:

$$TAW = RECIP(TA) + PHI.$$

At this point, the pilot could be provided with the information which would allow him to fly directly from his current position to the desired waypoint. Some supplementary calculations which simplify this task apply to both this solution and the one in the next section and are presented in Section 2.3.5.

As a measure of the computational complexity of the obtuse triangle approach to the R-Nav solution, the mathematical operations involved are listed below. The operations of addition, subtraction and RECIP are ignored since they are fast and relatively easy to implement in a microprocessor. The slant-range correction, which is common to both approaches is also ignored.

Multiply	11
Divide	1
Cosine	2
Arcsine	1
Square Root	2
Sine	1

2.3.4. The Vector Solution

A completely different approach to the R-Nav solution involves the use of two-dimensional vectors. The bearing/distance values for both the aircraft and waypoint positions are treated as RHO/THETA vectors in a polar coordinate system. Both vectors originate at the station which serves as the origin of the system. Normal vector addition may be used to solve the R-Nav problem.

As can be seen in Figure 6, when the waypoint vector is added to the reciprocal of the aircraft vector, the resulting vector is parallel to the vector from the aircraft to the waypoint. The resultant vector is also equal in magnitude to the vector from the aircraft to the waypoint.

Since the reciprocal of the aircraft vector is computed by simply adding or subtracting 180 degrees (the magnitude of the vector is always positive), this approach offers a somewhat simplified algorithm. The vector addition in itself does, however, present some problems. The approach which offers the most promise is to first convert the vectors

to a Cartesian coordinate system, form the sum, and then convert back to polar coordinates. The polar to rectangular conversion can be accomplished by using the formulas:

$$X = RHO \cdot COS(THETA)$$

$$Y = RHO \cdot SIN(THETA).$$

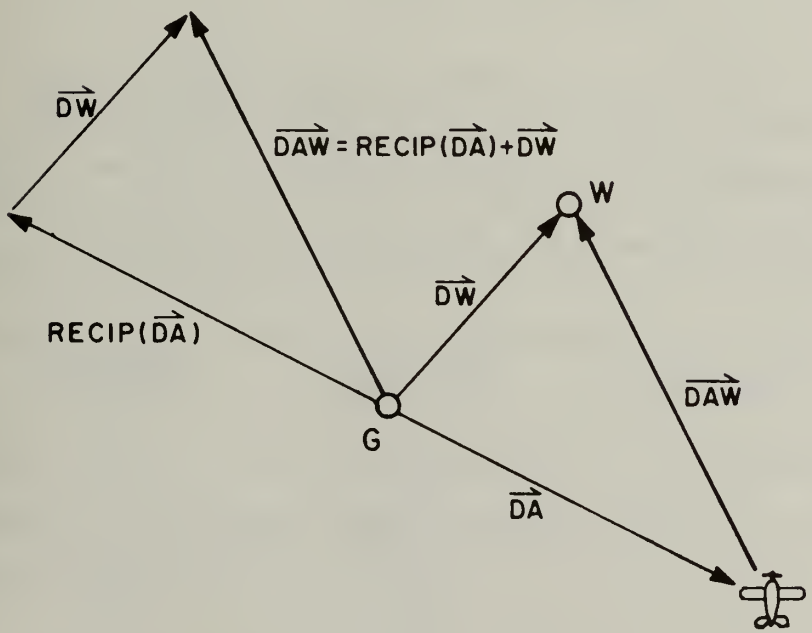


Figure 6. Vector solution of R-Nav geometry.

Since two vectors are involved, these operations are performed twice, once for each vector. Forming the sums:

$$X = X1 + X2$$

$$Y = Y1 + Y2$$

gives the Cartesian coordinates of the resultant vector. This may be converted to polar coordinates using the formulas [4]:

$$RHO = \sqrt{X^2 + Y^2}$$

$$THETA = TAN^{-1}(Y/X).$$

Once again, an ambiguity exists in the value of THETA. This ambiguity is reasonably easy to resolve, based on the signs of X and Y. The following table indicates the approach to resolution.

Sign of X	Sign of Y	Quadrant
+	+	I
-	+	II
-	-	III
+	-	IV

Table 1. Resolution of Ambiguity in ARCTAN Computation.

The computational complexity of this system is significantly less than that of the obtuse traingle solution. The operational requirements (again ignoring addition and subtraction) are:

Multiply	4
Divide	1
Square Root	1
Sine	2
Cosine	2
Arctangent	1

The sign comparisons are required to resolve the ARCTAN ambiguity, but this is a relatively fast procedure, even in a microprocessor. The reduction from 18 to 11 major mathematical operations seems to offer substantial improvement in the speed with which these computations can be performed. In addition to these obvious advantages, it should be noted that the conversion of the waypoint vector from polar to rectangular coordinates would need to be performed only when the waypoint is changed. This would be at infrequent intervals when compared to the time between positional computations.

2.3.5. Additional Computations

As will be seen in Chapter 3, one of the advantages of a microprocessor-based R-Nav System is to allow the microprocessor to assume some of the functions now incorporated into the navigational radios them-

selves. One of the functions which is built into the VOR bearing receiver which could be handled by the R-Nav System is the course deviation computation.

Conventional VOR receivers have provisions to allow the pilot to enter a course which he wishes to fly to the station. This is called the Omni Bearing Setting or OBS. Once this bearing is selected, the VOR receiver determines whether the selected course will take the aircraft to or from the station. This information is presented to the pilot in the form of two small flags, one of which is labeled TO, and the other FROM.

Once the pilot has chosen the OBS, the VOR receiver, using analog circuits, compares this bearing with the actual bearing from the station. The difference between the two is displayed to the pilot on an analog instrument called the Course Deviation Indicator or CDI. The CDI has a center null position, and hence indicates whether the pilot is left or right of the desired track. The magnitude of the deflection of the needle is directly proportional to the aircraft's deviation from the OBS. Older instruments present this information as the angle between the actual and the desired bearing. The distance-off-track for a given off-track angle will vary with the distance from the station. Newer instruments utilize the distance information from the DME and use an analog computation to produce the distance off track. In these systems, the pilot is usually given the option of choosing angle-off-track or distance-off-track. In

some systems, the scale of the distance-off-track indication is also selectable in two or more ranges. This allows the pilot to select relatively low resolution at long distances from the station, and finer resolution when near the station.

These functions of the VOR receiver can be readily performed by the R-Nav computer. The values can be computed for the course to a waypoint as well as directly to the station, which is the only computation performed by the VOR receiver. Navigation to the station is treated as the degenerate case of a waypoint at RHO = 0 and THETA = 0.

The TO/FROM, LEFT/RIGHT, and angle-off-course values are functions of the difference between the OBS and the actual bearing to the waypoint. (If this difference is less than 0, then 360 degrees is added.) This is shown in Figure 7. ANG is the adjusted difference between OBS and TAW. The TO/FROM function is undefined at the 90 and 270 degree points.

Once the off-track-angle has been computed, the off-track-distance can be found by computing:

$$OTD = DAW \cdot SIN(OTA).$$

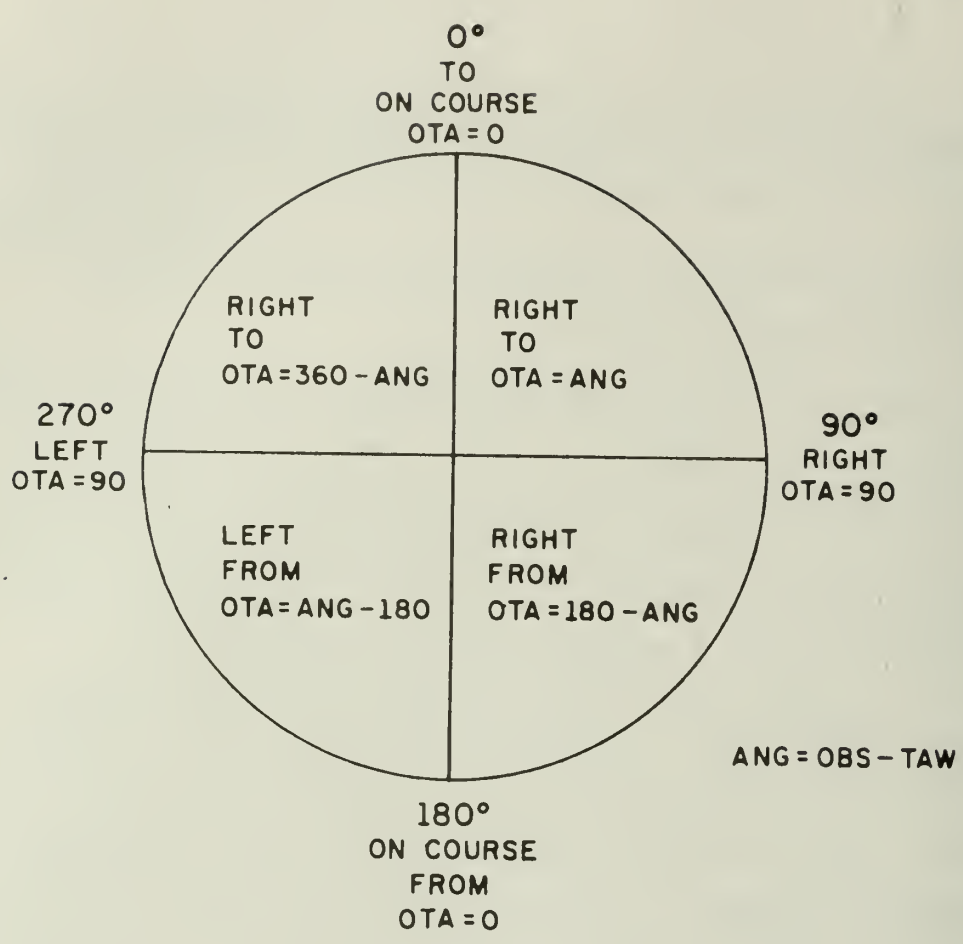


Figure 7. Computation to TO/FROM, LEFT/RIGHT and angle-off-track.

2.4. System Design Constraints

The preceding sections have defined the basic R-Nav computations. In addition to performing these computations, however, any R-Nav System must be designed within certain externally imposed constraints. The Federal Aviation Administration has established certain minimum operational characteristics which must be adhered to by any R-Nav System [9]. In

addition, the nature of the aircraft in which the equipment is to be installed will place limitations on certain aspects of the design. In the case of a microprocessor-based system, the functional characteristics of microprocessors will impose additional restrictions. These requirements and limitations must be treated as design constraints that must be incorporated into the final R-Nav System design.

2.4.1. Federal Aviation Administration Constraints

By far, the most significant constraint imposed by FAA is that of error tolerances. These restrictions are imposed in order to insure adequate traffic separation along the airways. The total error budget for R-Nav Systems is divided into various components such as VOR ground error, VOR airborne error, etc. [9]. The only error of concern here, however, is that error allocated to the airborne computer portion of the total R-Nav System.

The error budget for the R-Nav computer is divided into along-track error and cross-track error. While the other error components may vary with distance from the station and other factors, the maximum allowable computer error, both cross-track and along-track, is a constant 0.5 nautical miles [9]. The error specifications are predicated on 95 percent probability; that is, the errors must be within the specified limits 95 percent of the time. If it is assumed that the computer errors are normally distributed with a mean of zero, then the maximum standard deviation allowed can be computed, based on the fact that 95 percent of a normal distribution falls within $\pm$ 1.96 standard deviations of the mean [10]. Hence,

$$3.92\sigma^2 = 0.5$$

or,

$$\sigma^2 = 0.1276.$$

This assumption and some computed values for σ^2 will be discussed further in Chapter 3.

Several other FAA requirements are also specified, and while these are generally less significant than the error budget, they do constrain the design of an R-Nav System. Many of these constraints are obvious, but are included here for the sake of completeness.

The R-Nav System should give no operationally significant misleading information. This means, of course, that small errors in operation may be tolerated as long as they are not operationally significant. That is, a positional error which is small enough to allow the pilot to remain within his allocated airspace is acceptable. An error which would cause the aircraft to be operated outside the normal airway width would not be acceptable.

The R-Nav equipment should be able to maintain the specified accuracy within the full flight range of the aircraft in which it is installed. This includes not only the maximum airspeed of the aircraft, but different flight attitudes that would reasonably be expected of the aircraft.

The area navigation equipment should not be the source of objectional radio frequency signals which would interfere with the operation of other aircraft radios. Further, the R-Nav System should not be adversely affected by radio frequency emissions from other aircraft equipment. This would apply to the 1,200 MHz transmissions by the DME transmitter, as well as normal VHF and UHF communication transmitters.

The system should have provisions for alerting the crew in the event of a probable failure of major system function, or in the event of loss of inputs. In addition, the failure of airborne Area Navigation equipment should not interfere with the normal operation of required equipment connected to it. Note that this does not mean that all equipment must continue to function. Only the equipment required for the type of flight would have to remain operational. On a visual flight plan, technically, all equipment could fail along with the R-Nav, although this would not be desirable. For instrument operation, the equipment required to perform normal IFR operations would have to remain operational.

Provisions should be made for both ground and in-flight system testing. This requirement is reasonably easy to implement as far as the ground requirement is concerned. With the airborne test, it is not specified as to whether or not this test should interfere with normal navigation. At a minimum, an airborne test would result in a momentary interruption of navigational information. It is conceivable that it could also result in a loss of flight data which could take several minutes

to recover. It is assumed that an in-flight test during flight conditions which require the R-Nav for navigation would not be a normal procedure.

Waypoint horizontal position must be entered in increments of not more than 0.1 nautical miles, or 0.2 degrees. Provisions should be made to enable the pilot to verify correctness of all data inputs to the system.

Strangely enough, the correction of the slant-range error in the DME distance is not required below 18,000 feet [10]. For systems which do employ slant-range correction, the elevation of the VORTAC ground station must be entered in increments of not greater than 1,000 feet (enroute use). The waypoint altitude must be enetered in increment of not greater than 100 feet (enroute use) [9].

2.4.2. User Equipment Imposed Constraints

The flight characteristics of the aircraft which comprise the general aviation airfleet are very different than those of commercial or military aircraft. General aviation aircraft are small, light, predominately single-engined, and for the most part privately owned and operated. These aircraft are often operated by a single crew member, even during instrument conditions. These characteristics impose some constraints on the system design, while at the same time, mitigate other constraints, and allow certain simplifying assumptions.

The size of general aviation aircraft prohibit large, heavy navigation systems. Each pound of navigational equipment installed subtracts from the maximum allowable payload. These payloads are already restricted to crew, passengers, fuel, and a small amount of luggage. A one hundred pound navigation system would virtually prohibit luggage in many light aircraft. Therefore, the total R-Nav System must be kept as light as possible.

The electrical systems on general aviation aircraft are also limited when compared to the larger aircraft. A single 24 volt DC battery and generator must supply power to the entire aircraft, including communications radios, lights, etc. Power supply requirements for an R-Nav System for general aviation must be kept simple, and at the same time, power consumed must be kept low.

The pilots who operate general aviation aircraft are not generally professional pilots, and therefore, the R-Nav System must be easy to operate without extensive training. Further, since general aviation aircraft are often operated by a single pilot, in-flight demands on the pilot's time must be minimal. That is, the R-Nav System must reduce rather than increase the cockpit workload. At the same time, the system must still retain a high degree of responsiveness to the user requests for service.

The navigational equipment installed on most general aviation aircraft is limited. In keeping with providing an R-Nav System for this user population, it is unrealistic to assume the availability of certain

desirable input information for the system. For example, very few general aviation aircraft are equipped with Slaved Gyro Magnetic Compass Systems. Neither the normal magnetic compass nor a heading gyro (if installed) is sufficiently stable to provide reliable navigational information. Therefore, the system must be designed to operate in the absence of magnetic heading information.

Remote reading electronic altimeters are becoming more common in general aviation, but again, it is unrealistic to design a system predicated on the availability of this input. Similarly, the availability of airspeed information cannot be assumed.

While some aircraft are equipped with dual VORTAC radios, it would be undesirable to impose this requirement on aircraft desiring to utilize the R-Nav System. In those cases where dual installation is available, the R-Nav System could allow pilot selection on one or the other, and still function properly. The second VORTAC could be used as a backup input system.

In general, then, the microprocessor-based R-Nav System must be designed to operate without heading, airspeed or altitude inputs from the aircraft instruments. Of these three, only the altitude tends to remain constant for long enough periods of time to make pilot entry feasible.

The nature of general aviation aircraft does, however, allow some simplifying assumptions. Speed is one case in point. Even in light twin-engined aircraft, the speed seldom exceeds 240 knots. This low speed, of course, provides more time for computations than would be the case with higher speed aircraft.

These aircraft also generally tend to operate at lower altitudes. Due to a lack of oxygen equipment or cabin pressurization, most operations are below 10,000 feet. A maximum altitude assumption of 19,900 feet allows a wide margin for variations, and at the same time, simplifies some of the computations (slant-range error correction, for example).

The altitude also determines the reception distance for the VORTAC radios. This distance can be computed from the equation:

$$R = 1.23 \cdot \sqrt{HT} + 1.23 \cdot \sqrt{HR}$$

In this case, R is maximum reception distance in nautical miles, HT is height of transmitter antenna in feet, and HR is the altitude of the aircraft, in feet. At 10,000 feet altitude, R is approximately 122 nautical miles, if the station is assumed to be at zero elevation.

Due to a shortage of frequencies, however, this theoretical figure cannot be used by aircraft operating below 10,000 feet. With the exception of the 'H' class of high altitude VORTAC stations that are used above 18,000 feet, the maximum usable range is 100 nautical miles [11]. This range restriction is imposed to prevent reception interference between VORTAC stations operating on the same frequency.

The maximum usable radius defines the range of numbers which will have to be processed by the R-Nav System. With a range limitation of 100 nautical miles, the maximum distance figure which could be encountered would be 200 nautical miles, and this figure is highly unlikely. The expected distance between two points in a circle of radius R is [12]:

$$DE = \frac{128R}{45\Pi}$$

with R=100, this is approximately 90.5 miles. In general, the distances from aircraft to waypoint should be distributed about this value.

2.4.3. Microprocessor Imposted Constraints

There are several important limitations of microprocessors which must be considered in the design of a microprocessor-based R-Nav System. Any system that is designed without consideration of these limitations will result in a system which is, at best, marginal in performance.

Compared to either analog or discrete logic systems, microprocessors are slow. Gate time for standard Transistor Transistor Logic (TTL) discrete circuits are measured in nanoseconds. Even the faster N-Channel Metal Oxide Semiconductor (NMOS) microprocessors are much slower. A typical memory-to-register add instruction will require 2 to 10 microseconds in most microprocessors.

Additionally, microprocessors typically operate on short word lengths. Most microprocessors utilize words of from 4 to 16 bits, with 8 bits being the most common. A word length of 8 bits is only sufficient for values up to 256. Since the microprocessor R-Nav System will have to work with angles up to 359.9 degrees, with resolution to the nearest tenth of a degree, multiple word storage of data will be necessary. A 16 bit word would be sufficiently large to allow either binary or binary-coded-decimal (BCD) representation.

This multiple storage will require additional time for computation. In order to add two double word values together, two separate add operations will be necessary. One operation will add the low order words, and the second operation will add the high order words, plus any carry generated in the first step.

Microprocessors also have very limited arithmetic capabilities. There are currently no microprocessors with multiply or divide instructions. Multiplication will have to be accomplished with repeated addition, using the standard shift-and-add technique. Division is similarly accomplished by using repeated subtraction.

Most microprocessors perform only binary arithmetic. One, the INTEL 8080, has a decimal adjust instruction which will allow operation on BCD numbers, but at the expense of additional time. Only one microprocessor, the MOS Technology 6502 (and others in the 650X series), allows direct decimal arithmetic.

Microprocessors, in general, do not provide for signed numbers. If negative numbers are to be used, one of the data bits must be allocated for storing the sign. This, of course, reduces the range of values which may be stored. In certain operations, such as the shift-and-add multiply routine, additional operations may be required to compensate for signed numbers.

All the factors outlined above have one major impact on the operation of the R-Nav System. The time required to perform the navigation calculations will be longer than would be required with either

analog or discrete logic systems. This time factor becomes very important when it is applied to a navigational system. The position which is displayed by the R-Nav System is the position at which the aircraft was located at the time the system inputs were last read. During the time the navigational computations are taking place, the aircraft position continues to change. At a speed of 240 knots, the distance covered may be almost 0.07 nautical miles per second.

The time required for position computation may, therefore, be viewed as a computation error. This error must be included as a part of the total R-Nav error budget. The computation errors, plus the error induced by the time-for-consumption must not exceed the FAA limitations of 0.5 nautical miles.

3. APPROACH TO SYSTEM DESIGN

3.1. Design Philosophy

Many of the constraints affecting the design of the Microprocessor-based Area Navigation (MAN) System have been detailed in the preceding chapter. Even within these constraints, however, the designer has considerable latitude in many aspects of the design. Throughout the design of the MAN System, the basic philosophy has been to take full advantage of the capabilities of the microprocessor whenever possible, in order to reduce the amount of support hardware required. This approach serves to keep hardware costs down, while, at the same time, it reduces the power supply requirements.

It is not possible to eliminate all external hardware, but this philosophy has been pursued whenever possible. One example concerns the use of the keyboard for data input. Here, the design is implemented using the least expensive SPST push button switches available. Software routines within the microprocessor are then utilized to "de-bounce" the switches. This approach is made possible by the fact that keyboard entry takes place during periods when navigational computations are not being performed. Further, the key presses are widely spaced in time, relative to microprocessor speed. This extra processor time is utilized instead of external hardware.

Another instance of the application of this philosophy is the manner of obtaining the VOR and DME inputs to the system. These inputs could be obtained, using Analog-to-Digital converters, directly from the final output stages of the radios themselves. This would involve the use of the analog circuits within the radios to process the ground signals. A better approach would seem to be to obtain the raw data from the radios, as soon after the signal has been separated from the carrier as possible. The signals are then processed within the microprocessor.

This method of signal acquisition, which is detailed in Chapter 4, takes maximum advantage of the capabilities of the microprocessor. At the same time, it introduces a measure of redundancy into the total navigational system. The analog resolution portions of either the VOR or DME radio can fail, and the MAN System will still function normally. This is very much in keeping with the FAA requirement that the various systems should be as functionally independent as possible [9].

3.2. Mathematical Considerations

3.2.1. Number Representation

The range of input and output values which must be processed by the MAN System has been specified in the preceding chapter. Bearings will range from 0 to 359.9, and distances will not exceed 200.0 nautical miles. Negative values will not be encountered as either input or output, but could arise as intermediate values during mathematical calculations. Since the range of numbers is quite small, floating point representation is clearly not indicated. Fixed point representation is adequate to meet the needs of the system.

The choice between binary and decimal base is more difficult. Binary offers faster arithmetic, and generally, more efficient utilization of memory. It does, however, introduce the problem of decimal to binary conversion of all pilot directed inputs. It would be unrealistic to assume that the system user would be willing to input data in binary form.

Further, the FAA specifications of resolution to the nearest tenth of a mile introduces another problem [9]. Of the ten possible fractional decimal inputs, eight are irrational numbers in base 2. Errors would be induced in 80 percent of all user inputs, and these errors would propagate through the subsequent calculations. The final conversion of results back to decimal base for display would induce further errors. The detailed analysis of the effect of these errors is a task which it seems prudent to avoid. By choosing a decimal base in the beginning, error analysis can be restricted to the errors induced by the mathematical processes themselves.

The decimal representation choosen for the MAN System is the standard BCD format, using an 8421 weighting scheme. To cover the full range of input and output values, four such digits are required. This will require 16 bits for each number. Since the first digit will never exceed 3, there are two unused bits in the high order digit. These bits can be used with a tens-complement representation to handle negative numbers. The high two bits will be zero for the positive numbers 0 to 3. With tens-complement representation, at least one of them bits will be 1 for any negative number between -1 and -3.

This notation will make it possible for 16 bits to accommodate the full range of inputs and outputs of the system. Certain intermediate values involved in internal computations will require additional precision, but the range of numbers should not exceed those discussed.

3.2.2. The CORDIC Algorithm For Computing Functions

One of the major problems associated with utilizing a microprocessor to perform R-Nav computations involves the limited arithmetic capabilities of these microprocessors. Most microprocessors provide instructions for addition and subtraction only. In order to perform more complex functions, routines must be written in terms of these primitive operations.

The major computational problem for an R-Nav System is the coordinate system conversions as outlined in Chapter 2. If implemented as standard mathematical functions, these conversions would be very time-consuming, and require elaborate software routines for SINE, COSINE, ARCTAN, and square root. There is, fortunately, an alternative to this method of coordinate conversion. This is the CORDIC algorithm, developed by J. E. Volder [13]. This algorithm will provide polar to rectangular, or rectangular to polar, coordinate conversion, using an iterative technique that contains only the operations of shift, add and subtract. At the end of the iterations, a single multiplication step is required.

3.2.2.1. Introduction to the CORDIC Algorithm

The CORDIC algorithm performs coordinate conversion by performing a sequence of pseudo-rotations on a vector. When the original vector is aligned with the X axis, and is rotated through a given angle THETA, the result is the X and Y coordinates of a vector of the same magnitude, at an angle THETA. This is called the rotation mode, and in effect, performs polar to rectangular coordinate conversion.

The second mode, called the vectoring mode, begins with an X, Y coordinate pair. In this case, the vector associated with X and Y is rotated until the Y argument is zero. The total rotation is equal to the negative of the angle of the original vector, and the X argument represents the magnitude of the original vector.

In both modes, the pseudo-rotation is accomplished in a sequence of rotations, through angles of decreasing magnitude. In many respects, the process is similar to the familiar bisection method of root finding, where the desired root is bracketed into smaller and smaller intervals. The particular power of the CORDIC algorithm is based on the manner in which the successive angles of rotation are chosen.

Figure 8 shows a vector of magnitude RHO, and angle of Θ, and rectangular coordinates of X and Y. If this vector is rotated through an angle α, the rectangular coordinates of the new vector can be computed from:

$$X' = RHO \cdot COS(\Theta + \alpha)$$

$$Y' = RHO \cdot SIN(\Theta + \alpha).$$

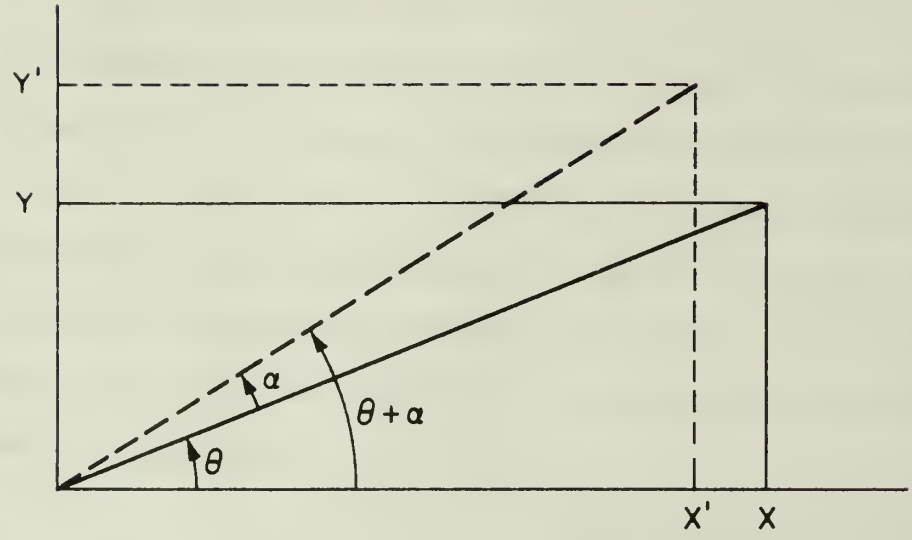


Figure 8. CORDIC rotation.

Working for the moment with the X' value, this can be rewritten as:

$$X' = RHO[COS\theta COS\alpha - SIN\theta SIN\alpha]$$

Now define:

$$X'' = X'/COS\alpha$$

And hence:

$$X'' = RHO[COS\Theta - SIN\Theta TAN\alpha].$$

Now since:

$$COS\Theta = X/RHO$$

$$SIN\Theta = Y/RHO.$$

Then:

$$X'' = X - Y \cdot TAN\alpha.$$

This shows that an approximation to the X value resulting from the rotation of a vector through an angle α can be computed from the original X and Y values and the tangent of the angle of rotation, α. This approximation is in error by a factor of $1/COS\alpha$. A similar development will result in an approximation for Y of:

$$Y'' = Y + X \cdot TAN\alpha.$$

A rotation in the opposite direction will produce the same formulas, with only a change in the sign of the second term. The major

significance of this development is that an approximation to the new X and Y values can be obtained, using only the operation of add, subtract, multiply and tangent.

A sequence of rotations through both positive and negative angles: $\alpha_1, \alpha_2, \ldots \alpha_n$ could be summarized by the formulas:

$$X''_i = X_{i-1} \mp Y_{i-1} \, TAN\alpha_i$$

$$Y''_i = Y_{i-1} \pm X_{i-1} \, TAN\alpha_i .$$

The error in the approximation increases with each iteration by a factor of $1/COS\alpha_i$. The sign used during a specific step would be determined by the direction of rotation.

In the application of this algorithm, the sequence of α_i's may be chosen in advance, and stored in the computer. The tangent operation will thus be reduced to a table-look-up. The given tangent α_i would be stored in the Ith row of the table. The only mathematically difficult step remaining is multiplication.

There are many sequences of α_i's which will rotate a vector through a given angle. If computational complexity were not a consideration, any sequence of α_i's which would achieve the desired rotation could be used. The only requirement would be that the sequence, both positive and negative, would have to have a sum equal to the maximum desired angle of rotation, within some acceptable tolerance.

Since computational complexity is a consideration, however, there is one specific sequence of α_i's which offers significant advantages over all others. This is the sequence of α_i's such that:

$$\alpha_i = \text{TAN}^{-1} (B^{-i}).$$

Where B is the base of the number system being used. Using this sequence greatly reduces the complexity of computing TANGENT α_i, since:

$$\text{TAN } \alpha_i = B^{-i}.$$

This will reduce the multiplication operation to a division by a power of the base. This operation is readily accomplished by shifting the multiplicand an appropriate number of places to the right.

Using this sequence of angles reduces the computation of the approximation to the X and Y coordinates to a sequence of table-look-ups, shifts, adds, and subtracts. This is, however, only an approximation to the X and Y values, and is in error by the factor:

$$K = \frac{1}{\prod_{i=1}^{n} \text{COS}\alpha_i}.$$

K is dependent on the values of $\text{COS}\alpha_i$, and the number of iterations performed during the rotation. If the number of iterations is fixed, and the sequence of α_i's has been chosen in advance, then K

becomes fixed. The correction of the approximation can be made after all iterations are completed. This requires only one multiplication step, by a constant, for each of X and Y. The iteration steps consist only of shift, add, and subtract.

3.2.2.2. Implementation of the CORDIC Rotation Algorithm

The original version of CORDIC was implemented with hardware in a computer called, appropriately enough, the COordinate Rotation DIgital Computer. The major components of the system were an X register, a Y register, an Angle register, and storage for the Angle constants. Adders and subtractors were provided, and provisions were included for the final multiplication step [13].

In the rotation mode, the vector magntidue is stored in the X register, the Y register is set to zero, and the angle is stored in the Angle register. At each step during the iteration, α_i is either added to or subtracted from the Angle register, with the object of reducing the Angle register to zero. At the same time, new approximations for X and Y are computed.

More specifically, during each iteration, one of two sets of operations is executed, depending on the value in the Angle register. If the Angle register is greater than zero, the steps are:

$$ANG = ANG - \alpha_i$$

$$X = X - Y \cdot B^{-1}$$

$$Y = Y + X \cdot B^{-i}.$$

If the Angle register is less than zero, then the operations are:

$$ANG = ANG + \alpha_i$$

$$X = X + Y \cdot B^{-i}$$

$$Y = Y - X \cdot B^{-1}$$

If the Angle register is exactly zero, then the correct rotation has been accomplished. It is, however, not advisable to terminate the iterations at this point, since the constant K depends on the number of iterations. Allowing a variable number of iterations may save a few iterations, but is is at the expense of a more complex algorithm, since K would no longer be a constant.

In general, if the Angle register contains zero and the desired number of iterations has not been completed, an arbitrary choice between addition or subtraction is made. In this manner, the number of iterations remains fixed, and the final multiplication step is simplified.

The end result of the process outlined above is the rotation of the original vector throug a negative angle which is equal in mangitude to the angle which was originally stored in the Angle register. This, in turn, produces the X and Y approximations associated with a vector of the magnitude of the original X value, but rotated to the desired angle. The X and Y values are then multiplied by K to obtain the corrected values.

The algorithms can be implemented in software with the use of an additional, temporary register. In the hardware version, two adder/subtractors were used to allow simultaneous computation of the X and Y values. This is impractical in software, and therefore, software implementation requires an additional register.

It should be noted that this algorithm may be used to perform the computations of SINE and COSINE functions. If the magnitude of the original vector is chosen as 1/K, then the values for X and Y represent the SIN and COSINE of the given angle, respectively, without the need for the final multiplication step. This algorithm is often used in hand calculators for this purpose [14].

The algorithm may be used with either degrees or radians, as long as the α_i's are in the same units. In fact, the original version was developed for use with binary fractions of 180 degrees [13]. It works equally well in all three cases.

The algorithm will work for angles from 0 to 90 degrees, if the iterations are started at I=0, and α_0=45 degrees. In practice, however, the first 45 degrees rotation is usually treated as a special case and iterations are begun at I=1. This approach is taken because the 45 degrees rotation is a particularly easy one. The X and Y registers are simply interchanged with sign changes as appropriate, depending on whether the 45 degree angle is added to, or subtracted from the Angle register.

In cases where angles larger than 90 degrees are anticipated, special case rotations of 90 and 180 degrees are implemented before iterations begin. An angle larger than 180 degrees may be represented as the equivalent negative angle which has an absolute value of less than 180 degrees.

3.2.2.3. Implementing the CORDIC Vectoring Mode

The vectoring mode is implemented in much the same manner as the rotation mode. The X and Y registers are set to the appropriate X and Y values, and the angle register is set to zero. Iterations proceed as outlined above, but with the the object of reducing the Y register to zero. That is, the choice of adding or subtracting α_i is chosen so as to reduce the magnitude of the Y register. If the X and Y registers are of different signs, the steps are:

$$ANG = ANG - \alpha_i$$
$$X = X - Y \cdot B^{-i}$$
$$Y = Y + X \cdot B^{-i}$$

If the X and Y registers are of the same sign, then the operations are chosen as:

$$ANG = ANG + \alpha_i$$
$$X = X + Y \cdot B^{-i}$$
$$Y = Y - X \cdot B^{-i}$$

As before if the Y register is zero, an arbitrary choice is made. The same special case rotations of 180 and 90 degrees are also required for angles greater than 90 degrees.

The effect of the vectoring mode of CORDIC is to produce an X register value which is:

$$XREG = \sqrt{X^2 + Y^2}.$$

Th Angle register contains a value:

$$ANG = TAN^{-1}(Y/X).$$

This is the equivalent of rectangular to polar conversion. The post-multiplication step is required to correct the value in the X register, but is not required for the Angle register.

3.2.2.4. The CORDIC Algorithm in Decimal

The CORDIC algorithm as described above works well in base 2. There is, however, a special problem which arises when attempting to use the routine with decimal numbers [15]. The first few α_i's (in degrees) are given in Table 2 for both decimal and binary bases (I=0 is omitted as a special case).

I	2^{-i}	10^{-i}	α_i Base 2	α_i Base 10
1	.5	.1	26.56	5.710
2	.25	.01	14.04	0.573
3	.125	.001	7.13	0.057
4	.0625	.0001	3.58	0.006

Table 2. CORDIC Constants.

It was previously stated that any sequence of α_i's could be used as long as the sequence would produce a sum which was at least as large as the desired angle of rotation. While the α_i's in binary can be seen to be approaching a sum near 45 degrees, it is obvious that the sum of the decimal α_i's will be much smaller. Regardless of the number of terms taken, the sum will be considerably less than 45 degrees.

The solution to this dilemma is to use each decimal α_i more than once. If, for example, α_1 is used nine times, the sum easily exceeds 45 degrees. This maintains the mathematical simplicity of the CORDIC algorithm with only a slight increase in the complexity of counting iterations.

In practice, decimal CORDIC routines are divided into major and minor iterative cycles. The α_i's are changed on each major iteration, while only the computational steps are performed during the minor iterations. As in binary CORDIC, it is advantageous to maintain a constant number of

iterations, and therefore, there are generally nine minor iterations for each major iteration [15]. The number of major iterations is chosen to achieve the accuracy desired.

The decimal CORDIC algorithm has the advantage of requiring fewer constants than the binary equivalent. Since each α_i is used several times, fewer constants are needed for a given number of iterations.

3.2.2.5. The CORDIC Algoirthm in R-Nav Compuations

The basis for the use of the CORDIC algorithm in the Man System is to perform the polar to rectangular and rectangular to polar coordinate conversions required by the system. The rotation mode is used twice, once to convert the vector to the waypoint to the X and Y coordinates of the waypoint, and again to obtain the X and Y coordinates of the reciprocal of the vector to the aircraft. These coordinates are then added to give the coordinates of the resultant vector. The vectoring mode is used to convert these X and Y coordinates back to polar coordinates. This gives the bearing and distance from the aircraft to the waypoint, and completes the basic R-Nav computation.

There are, in addition, two less obvious applications of the CORDIC algorithm to R-Nav. The rotation mode can be used to compute the distance-off-track once the angle-off-track has been found. The rotation mode of the algorithm is used with the angle-off-track and distance from the aircraft to the waypoint as the input vector. The Y value (after post-multiplication) then represents the distance-off-track.

The second application involves a minor and modification of the vectoring mode to make the correction for slant-range error in the DME input. In this case, the Y register and the Angle register are set to zero. The X register is set to the slant-range distance to the station. The altitude of the aircraft (in nautical miles) is pre-multiplied by 1/K for use as a test value. The vectoring mode is then executed with the object of increasing the Y register to the point where it is equal to the test value. After the iterations are complete, the X register is multiplied by K, which gives the horizontal distance to the station.

With the four applications outlined above, the CORDIC algorithm accomplishes all the major computations required of the R-Nav System. The question remains as to whether the microprocessor can perform the computations accurately enough and with sufficient speed to achieve results which are within the 0.5 nautical mile tolerance specified by FAA.

3.3. Initial Feasibility Studies

3.3.1. Computational Errors

The first step in answering the accuracy question was to determine the precision required in the CORDIC routine. While the inputs and outputs to the routine are fixed at four decimal digits, the α_i constants require more precision (see Table 2). This precision will determine the number of digits which will be required in the X, Y and Angle registers.

In order to determine precision required, the CORDIC algorithm (rotation mode only) was written in PL/C. The precision for the input values was fixed at (4,1). The precision of the X, Y, and Angle registers was varied, as was the precision of the α_i constants and K. Since it was expected that an 8-bit microprocessor would be used with multiple-word-length arithmetic, the precision was varied in steps of two from (4,1) to (10,7).

Monte Carlo simulation techniques were used to choose random waypoints and aircraft locations. Several runs were made, using approximately 1,000 cases per run. Throughout these computations, the fixed point PL/C subroutines, MULTIPLY and DIVIDE were used [16]. Double-precision floating-point values were computed simultaneously using normal trigonometric functions. The results of the two values were compared and statistics collected for each run. Preliminary results indicated that a precision of (8,5) would produce errors of less than 0.05 nautical miles.

To confirm these indications, the full algorithm for both modes of CORDIC system was then implemented using this procesion. A second set of runs was made to determine the mean and standard deviation to be expected with this precision. Again, double-precision floating-point computations were used as a basic for comparison.

During this series of runs, the aircraft position was fixed in each of 121 positions, representing ten nautical mile increments from 0 to 100 in both the X and Y directions. At each point, 1,000 random

altitudes and waypoint positions were chosen using Monte-Carlo techniques. This program is included in Appendix 1. Due to the size of the program, it was actually run eleven times, with each run obtaining results for eleven aircraft positions. An extract of these errors (rounded) is shown in Appendix 2, along with a summary of the combined errors. The errors were tested in this manner in order to insure that the errors were not positionally dependent. This assumption is supported by the data in Appendix 2.

After the distribution of the errors was established as described above, a CHI-SQUARE test was run on one set of errors. The results of this test are summarized in Appendix 3. The essence of the CHI-SQUARE test was that the errors are normally distributed about the mean given. This, in turn, means that 95 percent of the errors will be within 1.96 standard deviations of this mean. Since the mean is not zero, the errors are not symmetrically distributed about zero. This requires the consideration of the worst case for both X and Y. In the case of X, where the mean is negative, the worst case is:

$$MEAN_x - 1.96\ \sigma_x^2.$$

This is approximately -0.0135. For convenience, this will be used as both the upper and lower bounds, on X, since at least 95 percent of the errors in X will fall within the range 0 ± 0.0135.

In the case of Y, the worst case is:

$$MEAN_y + 1.96\ \sigma_y^2$$

which is approximately 0.0103. Again, at least 95 percent of the Y errors will fall within the range 0 ± 0.0103.

Since X has the larger error, a further generalization that 95 percent of all X and Y errors will fall within the range 0 ± 0.0135 is reasonable. These generalizations are for the sake of convenience in the discussion which follows. They are probably excessively pessimistic, but do guarantee that at least 95 percent of the errors fall within the range given.

3.3.2. Time-To-Compute Errors

The errors induced by the CORDIC routine and associated mathematics are only one part of the total system error. The second error arises due to the time delay between reading the system inputs and the display of computed position. The distance the aircraft travels during these computations is a source of positional error.

Using the mathematical error estimate from the preceding section and the FAA limits of 0.5 nautical miles, the maximum allowable value for this error (worst case) can be calculated from:

$$TTC = 0.5 - EC.$$

TTC must, therefore, be less than 0.4865 nautical miles in the case in question.

If the maximum speed of the aircraft is assumed to be 240 knots, then the maximum allowable time for computation can be determined. The worst case situation is considered, which is when the aircraft is traveling parallel to the X or Y axis. In this case, the maximum time for computation is:

$$MTC = TTC/240$$

or MTC must be less than or equal to about 7.3 seconds. Assuming an average time of four microseconds per instruction, this will allow approximately 1.8 million machine instructions to be executed between reading the inputs and displaying position, and still remain within FAA tolerances.

The results of this investigation indicate that a microprocessor could be used to perform the necessary R-Nav computations with sufficient speed and accuracy to make such a system feasible. Based on these preliminary results, the design of such a system was initiated.

3.4. Some Additional System Features

The majority of the system specifications have been discussed in this, or the preceding chapters. There are, however, some desirable, if not essential, features that should be included in the

system design. For example, accurate time is essential to all navigation. An interrupt-driven real time clock would cost little in terms of either hardware or software. Such a clock would provide the basis for computing the ground speed of the aircraft. The clock could also be utilized to compute Estimated Time of Arrival at the waypoint, based on ground speed and distance to the waypoint. This ETA feature is not implemented in the MAN System, but the availability of the clock would make the implementation of this function relatively simple.

Another desirable feature is the provision for storing and recalling several waypoints. This would allow the pilot to enter all his waypoints prior to take-off, and then recall them by entering a single waypoint number. This would greatly reduce in-flight cockpit workload. Provisions for ten waypoints are included in the MAN System, but this could be expanded if desired.

3.5. Problems Not Considered in the System Design

Since this thesis is a feasibility study, two very specialized problems are considered to be outside the scope of the research. The problem of shielding and noise immunity is an engineering problem which would have to be considered in the design of an operational system. This problem is closely related to packaging and installation, and hence, must be deferred until such time as a prototype system for in-flight testing is constructed.

The second problem concerns the method of utilizing the outputs of the MAN System. This is, again, to some extent dependent on installation. Further, the choice of a "best" method of presentation is a psychology problem. Therefore, the MAN System will simply output BCD numbers as they are computed. A sequence of seven-segment light emitting diode displays is provided solely as a means of testing the system. The final utilization of these outputs, including conversion to analog or video form, if desired, is left for further research.

4. SYSTEM DESIGN

4.1. Selection of Microprocessor

The constraints on the system design, as well as the philosophy attending this design, have been discussed in the preceding chapters. For the sake of completeness, those constraints which effect the choice of a microprocessor for the MAN System will be reviewed briefly at this point.

The microprocessor must be capable of processing words as long as 40 bits in order to handle the computations in the CORDIC algorithm. Decimal digits of the form (8,5) are employed in this routine. This requires either a bit-slice processor or one with multiple-word arithmetic capabilities.

The choice of BCD representation for internal storage requires that the processor have decimal arithmetic capabilities. This is perhaps the most restrictive of the constraints, since relatively few microprocessors have this capability.

The microprocessor must be reasonably fast, and if possible, utilize a single power supply with low power consumption. Support hardware requirements should not be excessive. In keeping with the philosophy of the design, the microprocessor should be readily available, and the price should be low.

The two microprocessors which come closest to meeting these specifications are the INTEL 8080 and the MOS Technology 650X series. Both these devices provide decimal and multiple-word-length arithmetic

capabilities. The 8080 uses a separate decimal adjust instruction after performing binary arithmetic. The 650X series has a single status bit which can be set to enable decimal addition and subtraction, using the normal add and subtract commands.

The 650X series, and more specifically the 6502, has some advantages over the 8080. It utilizes a single +5 Volt power supply. It also contains an internal oscillator, which requires only an external crystal to provide system clocking. The 6502 also has a simplier bus structure, since neither the data nor address bus is multiplexed.

On the other hand, the 6502 has a more limited instruction set than the 8080. The 8080 also has more registers and a larger stack. The 8080 currently sells for $37.50 and the 6502 for $25.00.

After considerable study, it was decided that either of these microprocessors was adequate for the task. The 6502 was finally chosen, based primarily on the fact that it requires substantially less external hardware to support the system. There is no implied claim that this is the best microprocessor for this, or any other application. It is a good microprocessor, and is adequate to serve as the basis of the MAN System.

4.2. The 6502 and the KIM-1 System

4.2.1. The MOS Technology 6502 Microprocessor

The 6502 is a standard 40-pin, N-Channel, Silicon Gate microprocessor, which performs 8-bit parallel data operations. It is capable of supporting an address space of 65,536 bytes of memory. It uses a single power supply, and an internally generated two-phase clock.

The 6502 utilizes a 16-bit unidirectional address bus for both memory access and input/output. Reading or writing an 8-bit word is identical to a memory fetch or store. An 8-bit bidirectional data bus is used for all data transfers to or from the microprocessor. A single R/W signal is used to control the direction of data transfer.

A Reset (RST) signal is used to initialize the system after power on. Two levels of interrupt are provided. The highest level is the Non Maskable Interrupt (NMI), which is always active. An Interrupt Request pin (IRQ) is also provided. Setting a single bit in the status register either enables or disables IRQ. Both interrupt input pins are open collector, which allows the "daisy-chaining" of interrupt requests. Interrupt polling is required if more than one interrupt is connected to the interrupt pin. Interrupt requests are serviced at the end of the current machine cycle.

Internally, the 6502 contains an Accumulator (A), two index registers (X and Y), a status register, and a stack pointer. The Accumulator is used for all arithmetic operations. The X and Y registers are used primarily for address indexing and loop counting. They are both more limited in scope of application than the Accumulator.

The stack pointer is used to maintain the current location of the stack. The stack is fixed at 256 bytes, and always uses Page 1 of memory (hexadecimal addresses 100 through 1FF). The stack grows downward toward 100. Stack overflow or underflow results in wraparound, with no indication of the occurrence. The only stack operations available to the programmer involve storing or recalling the Accumulator.

The status register contains the current status of the microprocessor. Seven bits are provided for:

Negative Result (N)

Overflow (V)

Break Command (B)

Decimal Mode (D)

Interrupt Disable (I)

Zero Result (Z)

Carry (C)

One bit of this 8-bit register is unused. For further information on the 6502 hardware, the reader is referred to the 6502 hardware manual [17].

The 6502 supports 55 separate instructions. Many of these instructions have several addressing modes, which brings the total number of instructions to some 146. There are 13 different addressing modes, and no single instruction supports all addressing modes. Some instructions support only one mode [18].

Page zero (hexadecimal addresses 00 through FF) has a special significance in the 6502. Many of the instructions allow a short form of addressing (8 bits instead of 16) to access page zero. When these instructions are used, the time for data acquisition is reduced, due to the fact that the high order byte of the instruction address is not needed. Memory is also saved in that the high order byte is not stored. The savings in storage is about one-third (two bytes per instruction versus three). The savings in time is the time required to fetch one byte, or about one μsec per instruction.

The addressing modes supported by the 6502 include:

Immediate (1 byte only)

Absolute

Zero Page

Accumulator

Implied

Relative

Indexed

Indirect (1 level only)

Indirect Indexed

Indexed Indirect

Zero Page Indexed

Absolute Indexed (X or Y register)

As mentioned previously, not all addressing modes are available with all instructions. For further information pertaining to the availability of addressing modes, the reader is referred to the 6502 programming manual [18].

4.2.2. The KIM-1 Microprocessor System

In addition to marketing the 650X microprocessor series, MOS Technology provides a family of support chips to complement the microprocessor. They also offer a complete 6502-based system on a single printed circuit board, called the KIM-1.

Early in this research, it was decided that considerable development time could be saved through the use of such a system. For this reason, the prototype MAN System was developed around the KIM-1 system. This eliminated many of the pitfalls of wiring and debugging such a system. It also eliminated the need to program several routines which have no real application in the MAN System.

The KIM-1 system consists of a 6502 microprocessor and the lowest 1,024 bytes of random access memory. It also has 2,048 bytes of read-only memory, which contains the KIM monitor and various service routines. These service routines contain facilities for examining and loading the contents of memory. There is also a serial teletype input/output routine, and routines for dumping and loading punched paper tape, via the teletype routine. A casette tape recorder interface is provided, as is an input keyboard and a six hexadecimal digit output display [19].

These features, while helpful in facilitating the development of the MAN System, would be of little value in the final version. One feature of the KIM-1 system which would be retained in the MAN System is contained in the 6530 read-only memory chips. This device, which is available separately, is called the Peripheral Interface Adapter or PIA.

The PIA contains two 8-bit bidirectional data ports. The direction of these data ports may be set, or changed, under program control. The direction is established by writing a 1 (output) or 0 (input) to the address of the PIA data direction register. Since there are two ports in each PIA, these data direction registers are labeled PADD and PBDD. The appropriate port can than be loaded into the

accumulator, or stored from the accumulator, by addressing the appropriate data register (PAD and PBD).

Since there are two 6530's on the KIM-1 board, there are two PIA's available. Only one PIA is used in the MAN System. This provides two ports, one of which is used for input and one for output. Due to the use of one of the 6530 pins as a chip select, there are actually only 7-bits available for use with the PBD register on the KIM-1 [19]. The unavailable bit is PB6, which was not needed in the MAN System.

Another feature of the PIA is the inclusion of a programmable interrupt timer. This timer may be loaded with any desired value between 0 and 255. The counter then counts down until zero is reached, at which point, an interrupt is generated. The counter can be programmed to count down on each clock cycle, every 8th clock cycle, every 64th clock cycle, or every 1024th clock cycle. When the interval timer is used, the interrupt is generated on Pin PB7 of the PIA. PB7 then becomes unavailable for use as a data bit, since it must be connected to the IRQ line of the 6502 [17].

The PIA's are used in the MAN System to facilitate passing control signals to and from the external logic. The PIA's offer the advantage of requiring no external latches when these control signals are output. PIA pins programmed as output utilize an internal latch to hold the signal until it is changed. PIA pins programmed as inputs do, however, require external latches if the input signals are of a transient nature.

The interrupt timer provides the basis for the real-time system clock. The clock could have been realized with external logic, but the use of the interrupt timer provided this clock with no external hardware.

As mentioned previously, the KIM-1 hardware for TTY and casette tape interface would be discarded in an operational MAN System. The software monitor would also be discarded, being replaced with read-only memory containing the software for the MAN System. At least one PIA, with its associated interval timer would be retained. Since one PIA is included with each 6530 ROM, it could be obtained with the ROM. If some other type of ROM is used, the PIA is available separately, on a single chip, from MOS Technology.

4.3. Expansion of the KIM-1 System

The first step in adapting the KIM-1 system to the MAN configuration was to add address and data bus buffers. The 6502 can drive only one TTL load. Therefore, buffers are required to provide the external drive capability needed. Type 8T97 chips were used for the address buffers. These gates have a fanout of approximately 30 TTL loads. The gates have tri-state outputs, but since the address bus does not require this feature, the gates are permanently enabled. This is accomplished by connecting the enable input to ground (see Figure 9).

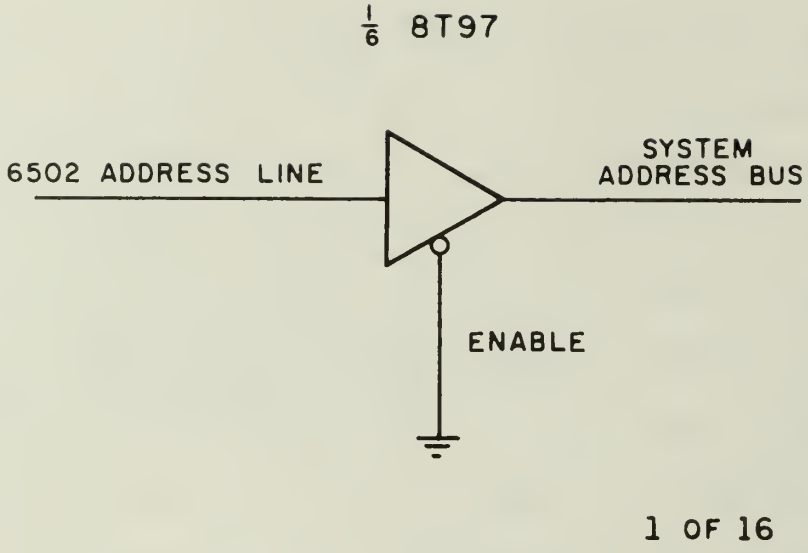


Figure 9. Address bus buffer.

The same 8T97's were also used to buffer the data bus. Since the data bus is bidirectional, two gates from the 8T97 chip were used for each data line. These were wired back to back with only one direction being enabled at any one time. The enabling of these gates is shown in Figure 10.

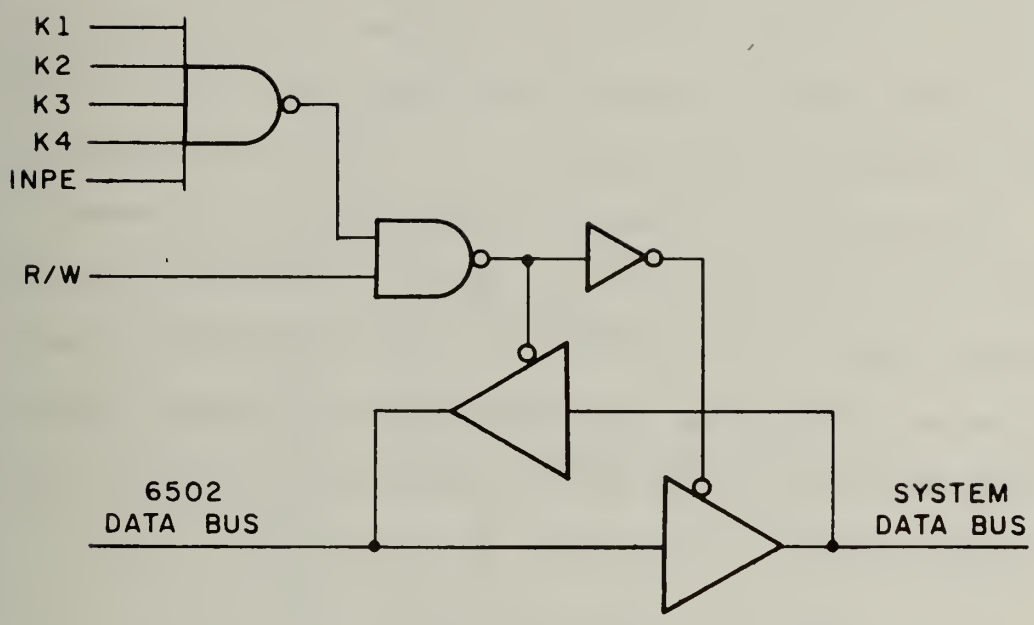


Figure 10. Data bus buffer.

The 1024 bytes of RAM provided with the KIM-1 was insufficient for the needs of the MAN System. An additional 4096 bytes of RAM were added. Since the KIM-1 system provides address decoding for the first 8192 bytes of memory, the additional memory was added in the second through the fifth 1024 byte positions of memory. This memory is enabled by the K1 through K4 signals provided by the KIM-1 decoder. Since the decoder has open collector outputs, pull-up resistors were also added.

The extra memory is composed of thirty-two 2102-1 memory chips with an access time of 500 nanoseconds. This speed is sufficient to keep pace with the 1 MHz clock supplied with the KIM-1. Each 1024 byte block of these chips is enabled by one of the address decoder outputs from the KIM-1. To prevent overloading the address bus, each of the lower ten address lines was inverted on the memory card. In this manner, the entire 4096 bytes of memory appears as one load to the address bus.

The data bus must be enabled for input whenever this memory is accessed and the R/W signal is high, indicating that a memory read is taking place. Except during this time, or during an input device read, the data bus may remain in the output mode. The data bus input enable is active (low) when either of the decoder outputs K1 through K4 is low and the R/W signal is high. This is, then:

$$\overline{BEINP} = (\overline{K1} + \overline{K2} + \overline{K3} + \overline{K4}) \cdot RW$$

The bus enable for output is simply the complement of this signal. The implementation of this signal is shown in Figure 10.

After the 4096 bytes of memory was added, provisions were made for input/output devices. In order to limit the address space to 8K, the I/O devices were assigned addresses in the lower portion of the sxith 1024 byte block of the address space (see Figure 11). The remaining signal in Figure 10 is related to the input devices. Since the KIM-1 system uses the upper portion of the sixth block of the address space for

the PIA's and some RAM, address decoding was necessary to enable these devices. At the time one of these input device is enabled, the data bus must also be enabled for input. This is the INPE signal in Figure 10.

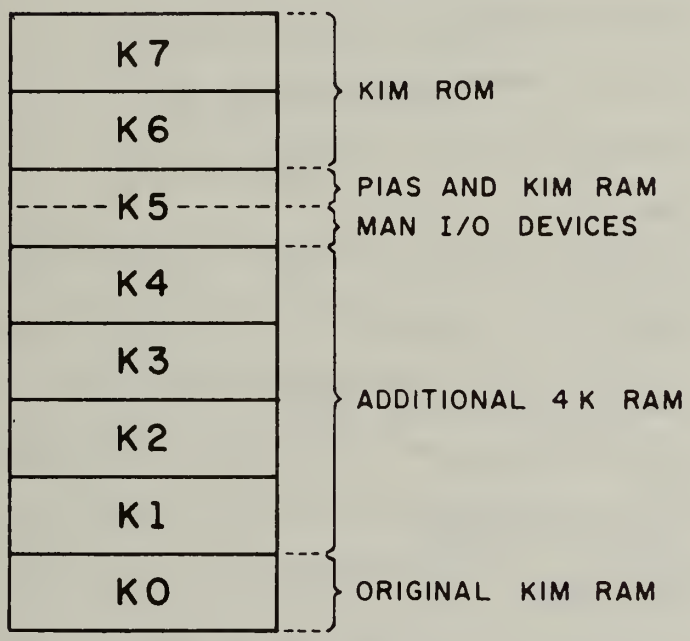


Figure 11. KIM-1/MAN address space allocation.

4.4. The MAN System

4.4.1. Output from the MAN System

While the exact form of output from the MAN System is not considered to be of direct concern in this thesis, some method for displaying results is required for system testing. The minimum information which is considered unable for such testing would be:

Waypoint number

Waypoint bearing

Waypoint frequency

Altitude

Time

Bearing to waypoint

Distance to waypoint

Omni Bearing Setting (OBS)

Angle off course

Distance off course

Ground speed

All these consist of four decimal digits except time (six digits), waypoint number (two digits), and altitude (three digits). In addition to these numeric outputs, some additional status information is required. This consists of the TO/FROM ambiguity function and a LEFT/RIGHT indication for angle and distance off course. Since this information is inherently binary (on or off) in nature, a single 8-bit latch is sufficient to contain this data.

While not absolutely essential, another set of status signals is considered desirable for output testing. These signals would be required in any operating system, and are also binary in nature. They could also be stored in a single 8-bit latch.

Station Passage Alert

Waypoint Passage Alert

Bearing Input Unreliable

Distance Input Unreliable

Dead Reckoning Mode

Ground Speed Not Established

The first set of signals mentioned above are normal system indications, and one each of the TO/FROM, LEFT/RIGHT pairs is normally on. An exception arises when the aircraft is exactly on course, and is neither left nor right of course. Another exception is when the aircraft is on a bearing which is perpendicular to the desired course (OBS). In this case, the TO/FROM function is undefined. In the MAN System, these four signals are referred to as the Low Status latch.

The second set of signals represent more or less unusual situations of which the user should be aware. In most normal situations, all these signals will be off. These signals are associated with the High Status latch.

Counting each two decimal digits as one output, there are a total of 23.5 numeric outputs from the system. For convenience, and to reduce external hardware requirements, it was decided to use an external register file to store these decimal outputs. This allows multiplexing the seven segment decoders and thus reduces hardware. The register file is constructed from three rows of four 74170 chips. Each chip will store four 4-bit numbers. By arranging the file in this manner, each row can be thought of as containing the equivalent of four 16-bit numbers. This is equivalent to four, 4-digit decimal numbers. The 74170 registers were chosen because they have separate read and write address lines. The arrangment of this register file and the attendant circuits is shown in Appendix 4.

The advantage of the separate read and write address lines is to allow the continuous multiplexing of the data outputs, even while data is being written into the register. The twleve rows of digits are multilexed through four 7448 seven segment decoders which drive the 47 seven segment displays. (The high order digit of the altitude is stored, but not displayed.) A modulo 12 counter is used to control the multiplexing of the registers, and simultaneously, to turn on the appropriate four display cathodes.

It was originally planned to drive the displays directly from the 7448's, but this was found to be impractical. Since each display is on only 1/12th of the time, the 7448's do not provide sufficient drive current to make the displays usable in a well-lit room. For this reason, an NPN transistor was added to each output of the 7448 decoders to provide

drive current. Type 75492 multiplexed LED digit drivers were added to the cathode circuit to sink the drive current for four digits at a time. Each 75492 is capable of sinking 250 ma. of current.

It was initially intended to use the system clock to drive the multiplex counter at 1 MHz. After trying this, it was determined that the 75492's were too slow to operate at this speed. A single 555 timer was wired as a 20 KHz clock to drive the multiplexer counter at a speed in keeping with the capabilities of the rest of the circuit.

The two status latches were constructed from 74174 latches, which drive light-emitting diodes directly. The address decoders which drive these latches, and the register file are shown in Figure 12. The utilization of these decoded signals in the register file is shown in Appendix 4.

4.4.2. Inputs to the MAN System

The address decoding for the five major inputs to the MAN System are shown in Figure 12. These inputs consist of two 8-bit inputs for the VOR bearing information, two 8-bit inputs for DME distance information, and one 8-bit input for pilot service requests. Numeric data entered by the pilot is handled through the PIA.

4.4.2.1. Acquisition of VOR Input

The acquisition of the VOR bearing information may be treated as a counting process. A counter that starts when the reference signal crosses zero (positive) and stops when the phase-varying signal crosses zero will contain a value that is proportional to the bearing from the

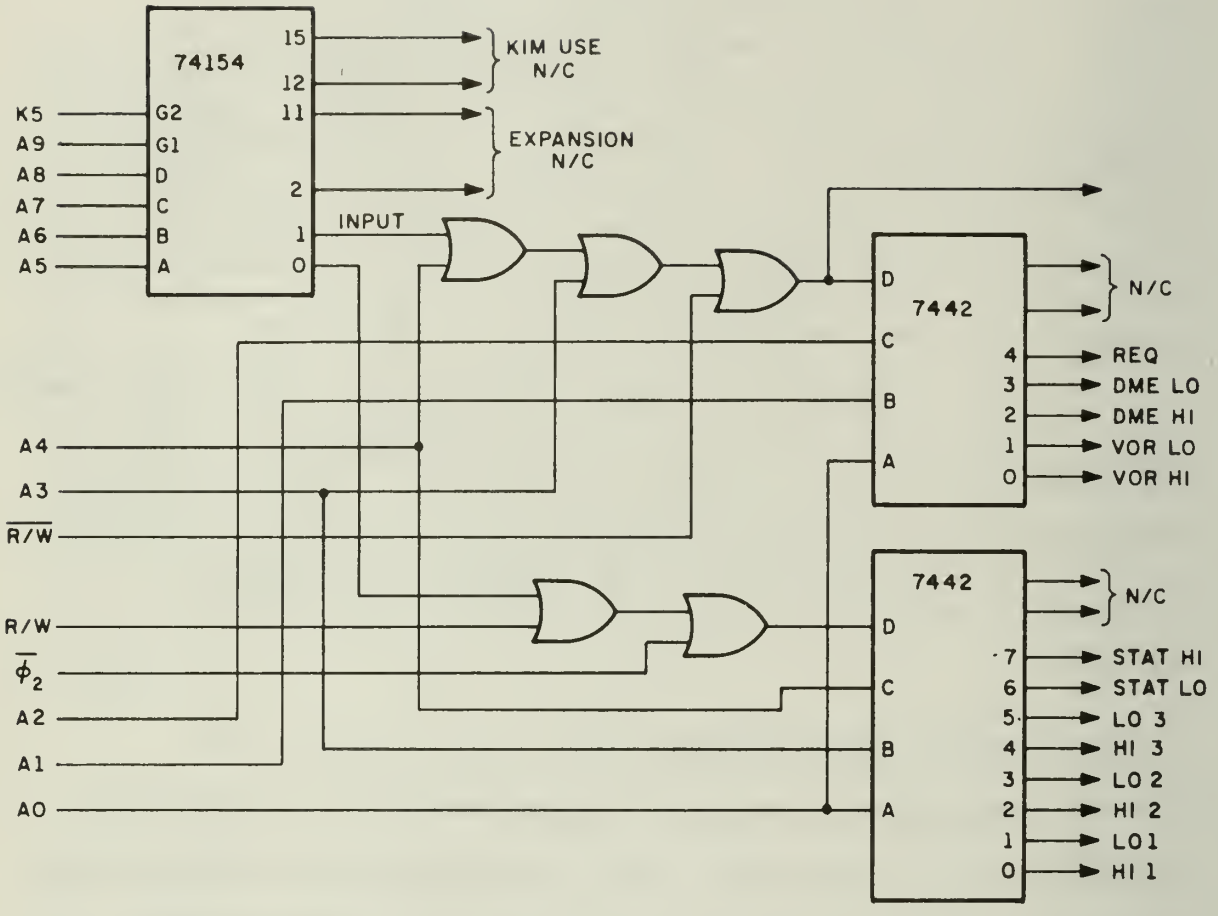


Figure 12. Input/Output Address Decoding.

station. If this counter is a BCD counter driven at the proper rate, it will contain the bearing from the station. Such an arrangement is shown in Figure 13.

The phase-varying signal transverses 360 degrees thirty times per second, or 10800 degrees per second. If the clock driving the circuitry shown in Figure 13 has a frequency of 10.8 KHz, then the counter will count in degrees. If resolution to the nearest thenth of a degree is required, then the clock can be driven at the rate of 108.0 KHz.

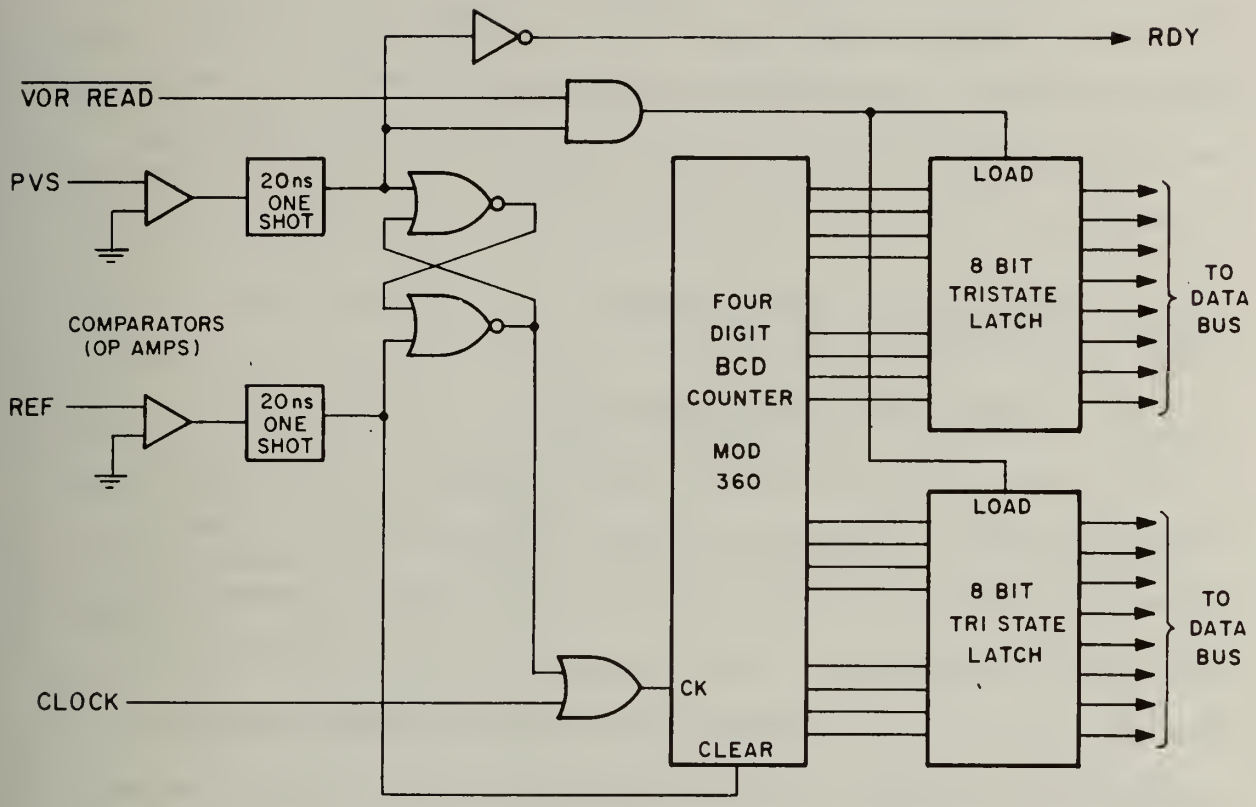


Figure 13. VOR bearing acquisition circuit.

The reference and phase-varying signals should be extracted from the VOR radios immediately after detection. If necessary, operational amplifiers may be used between the VOR radio and the comparators both to avoid loading the receiver circuits, and to adjust the inputs to the comparators to the proper level.

This particular circuit was not constructed for the MAN System, since no source of phase-varying signals was available for testing. The circuit is simple enough to present no implementation problems. In the MAN System, four thumb-wheel switches which putput BCD code are used to simulate the counter portion of this circuit. The signal is latched into a 74173 latch with the system clock.

4.4.2.2. Acquisition of the DME Input

4.4.2.2.1. Operational Signal Acquisition

The acquisition of the DME input can also be treated as a counting problem. A counter which begins counting when the airborne transmitter sends out a pulse, and stops counting when a response is received will contain a count which is proportional to the distance from the aircraft to the station. To allow for the known delay in response from the ground station, the counter should not start counting until 50 μsec after the transmitter has interrogated the ground station.

Radio signals travel round trip in air on a standard day at the rate of one nautical mile per 12.361 μsec [7]. In order to have a tenth of a mile resolution, the clock driving the counter would have to have a period of 1.2361 μsec. This corresponds to a frequency of approximately 808996 Hz.

In the MAN System, the microprocessor will control the DME transmitter, which will interrogate the ground station upon request from the microprocessor. This request signal can be used to trigger a 50 μsec monostable multivibrator which will clear the counter. Counting would then start 50 μsec after the interrogation pulse is transmitted.

A single signal from the DME receiver would be used to signal the micrprocessor that a response had been received. The same signal would load the contents of the counter into a latch for reading by the microprocessor. An operational amplifier could be used for signal level adjustment, if required. The amplifier could place negligible load on the receiver circuitry.

Since there is a 100 μsec delay between ground station transmissions, there is sufficient time for the microprocessor to read and store this latch before another response can be received. A signal tied directly from a distance counter to a microprocessor input would be used to signal when the maximum allowable range of returns has been reached.

4.4.2.2.2. DME Simulation in the MAN System

The arrival of random signals from the ground station makes the operation of this input considerably more complex than was the case with the VOR signal. For this reason, it was decided that a complete circuit to simulate this input was required. This simulator circuit is detailed in Appendix 5.

In order to simulate the random responses of the ground station, a random sequence counter was constructed. This is essentially a 24-bit shift register which uses an exclusive OR to feed bits 19 and 24 back to the input of the register. Such a counter will progress through 16,766,977 pseudo-random states before repeating [20].

In the simulator, the counter is started at 000.05 (decimal) by using an additional J-K flip-flop between the clock and the four digit BCD counter. Starting the counter with this bit set, and using a clock at double the previously mentioned frequency (1.617992 Mhz) has the effect of rounding the distance up to the nearest tenth of a mile.

Using the clock frequency of 1.617992 MHz and the maximum random count sequence of 16,766,977 gives a random count sequence which repeats only every 10.4 seconds. Since it takes only approximately 2472 μsec to cover a 200 mile range of DME returns, and since the requests for interrogation are under microprocessor control, the counter is, for all practical purposes, random.

A certain amount of care must be exercised with this counter. Should it over reach a state of all zeros, it will remain in that state indefintely. It cannot reach that state from any state in the normal count sequence. The only possible way to reach this state, barring a hardware failure, is during initial power up. For that reason, the reset signal is ORed to the input of the shift register. Since the 1.6 MHz clock runs continuously, this insures that the all zero state is not reached during power up.

In order to simulate the operation of the DME ground station, it is desired that the DME simulator produce approximately 2,700 random pulses per second. In a counter which passes through all possible states, any single bit will be on exactly 50 percent of the time. Any two bits will be on 25 percent of the time, and any N bits will be on

$2^{-N} \cdot 100$ percent of the time. In order to obtain 2,700 pulses per second, it is necessary to choose N such that:

$$2^{N} \cdot f = 2700.$$

Solving this equation for N gives a value of -9.227. Since N must be an integer, a value of -9 is used, which will produce a pulse rate of 3160 pulses per second. This is an error toward the more critical side, and is acceptable.

In order to obtain 3,160 random pulses per second, it is only necessary to AND any nine bits of the counter. The output of this gate will then be 1 in a random pattern which will be approximately 3,160 times per second. This circuit, shown in Appendix 5, generates the random pulses simulating the normal operation of the ground station.

To simulate the 100 μsec delay between responses, a monostable multivibrator with an on-time of 100 μsec is used. Whenever a random pulse is generated by the counter circuit, this one-shot is triggered. The output of this gate is fed to the microprocessor as the DME response received signal (DME READY). At the same time, this signal is gated back to the trigger of the one-shot to disable further inputs until the 100 μsec time period has elapsed. System software will insure that the 100 μsec pulse is treated as a single response.

In order to simulate the responses to the MANS equipped aircraft's interrogations, a set of four thumb-wheel switches was installed on the MANS front panel. Four 7485 comparators are used to compare the value set on these switches with the value on the distance counter. The

A=B output of the comparator is then ORed with the random pulse input to the 100 μsec monostable multivibrator. When the value on the counter is equal to the value on the switches, a signal is sent to the one-shot. Provided that the one-shot has not been triggered in the preceding 100 μsec, this will generate a DME response received signal. If the one-shot has been triggered in the preceding 100 μsec, this input will be ignored, and the response will not be recognized by the microprocessor.

This system very closely approximates the behavior of an actual DME system. A switch on the front panel of the MANS allows the A=B input to the comparator to be set to either zero or one. When this input is set to zero, the output of the comparator is always zero, and all pulses generated are random. This simulates an inoperative or out-of-range ground DME station.

The 2's-bit of the high order digit on the distance counter is used to signal the end of the DME operation (TIMEOUT). When this digit comes on, the microprocessor is programmed to discontinue waiting for DME responses and proceed with other operations.

The three signals outlined above are passed between the microprocessor and the DME simulator through the PIA. The DME GO signal (output) which clears the counter through the 50 μsec one-shot, is on pin PA3. The DME READY signal (input) which indicates the reception of an interrogation response is on pin PB5. The TIMEOUT signal (input) which signals the end of the search is on pin PB4. The utilization of these signals is described in Chapter 5 under System Software.

4.4.2.3. Pilot Directed Input

The input from the MAN System operator can be divided into two classes of inputs which are:

a. A request for a specific service.

b. Entry of numeric data.

An example of this division would be when the pilot wishes to input a change of altitude. The micrprocessor must first recognize that a request for service has been made. The type of request must be identified as a change of altitude. Finally, the new altitude information must be entered in numeric form.

None of the requests for service are of the nature that even a few seconds delay between request and service is critical. Certainly a one-second delay would be quite acceptable from the pilot's standpoint. Further, halting the processor during the course of positional computations would not appear advisable. If the numeric data entry required several seconds, it would be somewhat risky to resume computations with at least part of the data several seconds old.

These considerations lead to a polling approach to the request entry procedure. Rather than allow the request to interrupt the processor, requests are latched into a request register at the time of entry. Software routines examine this register at the end of each computation cycle to determine if a request has been entered. If there is no request, computations are resumed. Upon detection of a request entry, software routines service the request. After servicing the request, the request latch is cleared and computation is resumed.

4.4.2.3.1. Request Entry

There are six specific service request recognized by the MAN System. These requests are:

Waypoint Entry

Altitude Entry

Omni Bearing Set Entry

Test

Run

Stop

The meaning and software processing of each of these requests is covered in detail in Chapter 5.

The hardware implementation of the request latch is realized with two 74173 tri-state latches connected to the data bus. The clear input to these latches is connected to pin PA3 of the PIA. The inputs to the latch are connected to the input switches. These switches are normally open, which causes the inputs to the latches to be normally high. Closing a switch places a zero input on one of the six inputs to the latch (two inputs are unused). The same signal triggers a monostable multivibrator (74121) which generates the clock signal to latch the inputs into the flip-flops. This monostable is set to approximately one-half second to eliminate switch bounce.

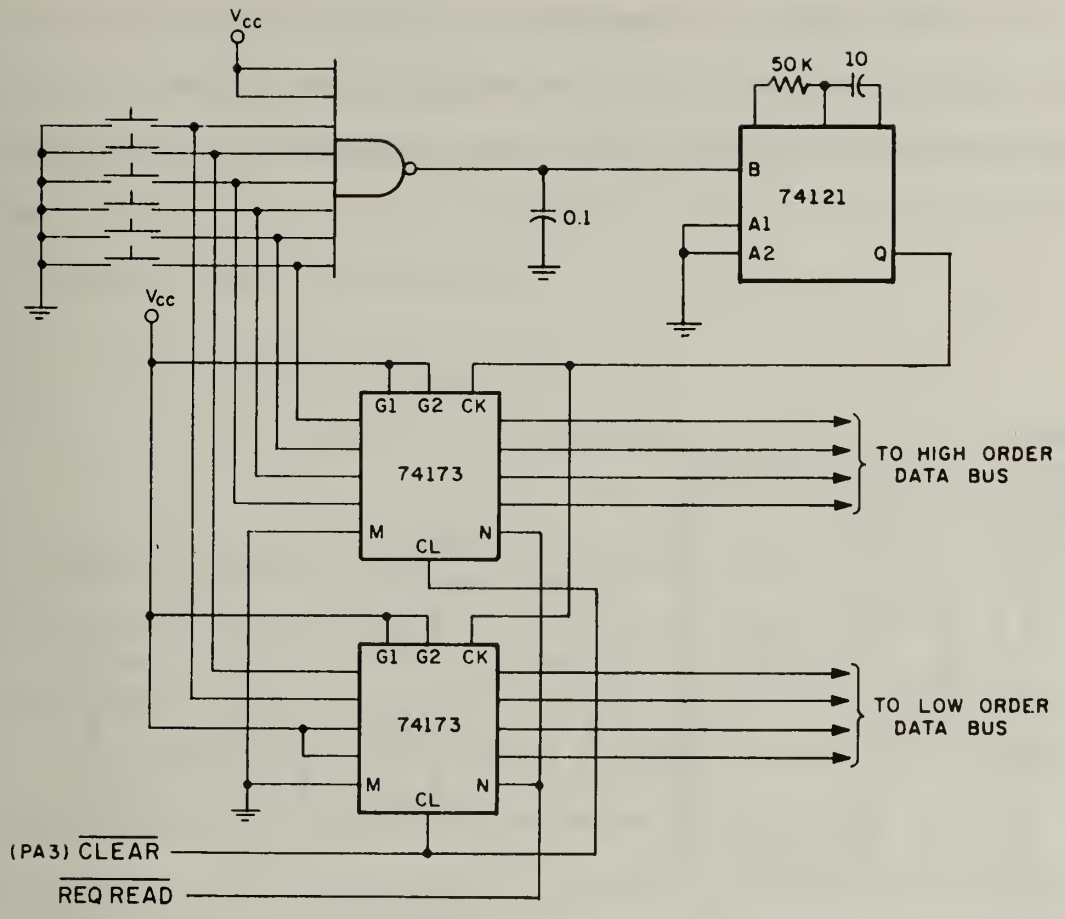


Figure 14. Request latch circuit.

The latch is read under program control to determine the entry of a request. When a request is entered, all bits in the latch will be one except the one corresponding to the depressed switch. Software can then determine the type of request which has been entered.

4.4.2.3.2. Numeric Data Entry

All service requests except STOP, RUN, and TEST require the entry of numeric data. This data is entered through a separate data entry keyboard. The data entry keyboard contains the normal ten decimal digits, a Clear Entry (CE) and a SET key.

The switches on this keyboard are arranged in a four row by three column matrix. The control and recognition of data inputs is accomplished through the PIA. The circuit is detailed in Figure 15.

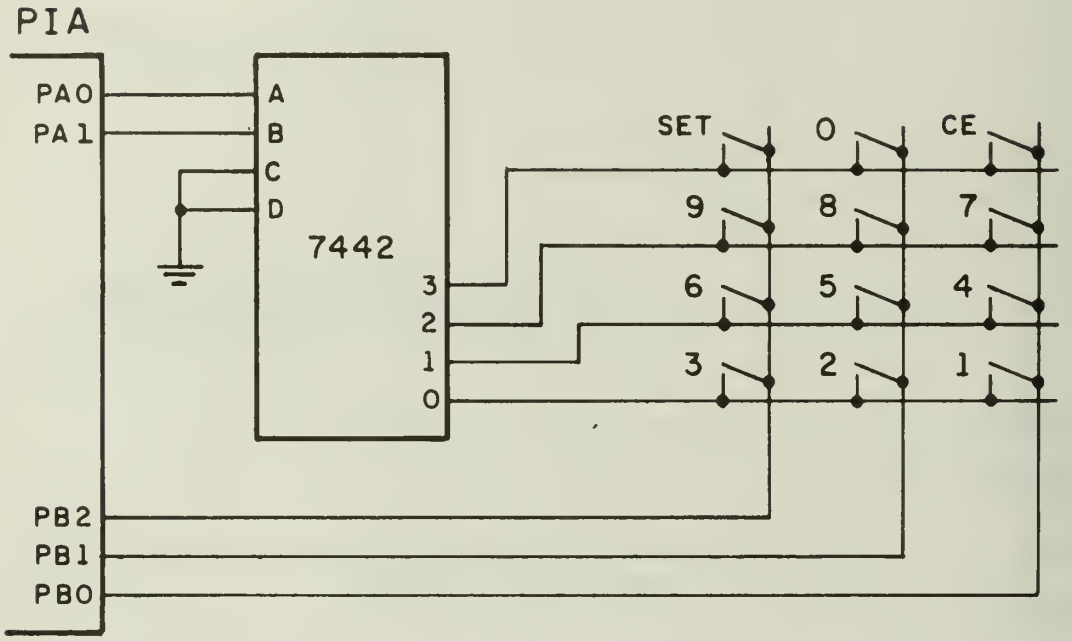


Figure 15. Numeric data entry circuit.

The PIA output pins PA0 and PA1 are used to output a numeric keyboard row code. This code is fed to a one of four decoder, which will apply a low signal to one side of each switch in one of the four columns. The other side of all switches in a single row are connected, respectively,

to PB0, PB1 and PB2 of the PIA. These inputs are read under program control. The occurrence of a zero on any of these pins indicates the depression of a key within the column which is active. Software is used to determine which key was depressed, and to convert the input signal to the appropriate representation.

4.5. System Failure Indication

Since the MAN System is essentially dependent upon the operation of the microprocessor to provide all navigational information, the loss of the microprocessor function would be quite serious. Since the VOR and DME radios are to be retained in their current form, the loss of the microprocessor would still leave the pilot with VORTAC capability.

Because the microprocessor could fail with all the outputs active, it might be several minutes before such a failure would be detected by the pilot. This could result in a hazardous situation, and therefore, a system failure indicator would be essential in any operational MAN System. This system is not implemented in the Prototype MAN System.

There are a number of ways to implement such an indicator, but the easiest to construct involves the use of a monostable multivibrator with an on time of a few seconds. This multivibrator would be triggered by a signal written periodically from within the system software. One of the PIA output pins could be used for this purpose. As long as the monostable receives periodic trigger signals, the output would remain high, and the failure light would remain off. Failure of the system would stop the periodic trigger signals, and after a short delay, result in the illumination of the failure light. A 74121 monostable multivibrator would

be used since this type gate is edge-triggered. System failure would be effected regardless of the state of the trigger signal at the time of failure. For additional redundancy, the failure indicator should have an independent power supply.

The system failure indicator would also be connected to a selector within the DME transmitter. During MAN System operation, it is necessary to trigger DME interrogations from the microprocessor. Should the microprocessor become inoperative, the control of interrogation triggering would have to revert to the random timing circuits within the DME radio. A selector would disable the DME triggering circuits as long as the microprocessor is operating.

5. SYSTEM SOFTWARE

5.1. Introduction

5.1.1. The General Approach

The general approach to the development of the MAN System software was the extensive use of subroutines. Whenever possible, the subroutines were generalized for use in several applications. The subroutines were also designed in a hierarchial fashion. The lowest level of subroutine usually performs some very specific task in detail. Higher level subroutines may utilize several lower level subroutines to perform a broader system function.

A good example of this hierarchy is the entry of numeric data from the keyboard. The lowest level of subroutine is the GETKEY routine which detects and decodes the depression of a single data entry key. One level higher is the GETBRG routine, which utilizes the GETKEY routine four times to allow the entry of a bearing. GETBRG verifies the validity of the data entered (0-359.9 degrees) and signals if an error is detected.

At the third level is the GETFOR routine used whenever four digit entry is expected. This routine establishes what data is to be entered, and where it is to be stored. In the case of a bearing, the GETBRG routine would be called from GETFOR.

The highest level is the program proper, which performs the normal R-Nav computations, and polls the request latch. Upon detection of a request requiring a four digit entry, the GETFOR routine is called.

The number of levels may vary in different parts of the program, but the basic concept is utilized throughout. This approach was taken in order to simplify the basic routines to the point where testing was relatively easy. It also provides a reasonable measure of efficient memory utilization, although at the expense of decreased speed. The subroutine calls are expensive in terms of execution time, but, as will be seen in Chapter 6, not prohibitively so.

5.1.2. General Program Description

At the highest level, the program consists of three major components: Restart sequence, Data entry sequence, and Run sequence. The restart sequence is utilized to establish the configuration of the machine, including the status of the PIA ports. In the MAN prototype, the restart sequence is entered via the KIM-1 monitor, rather than through the use of the RESET signal.

The data entry sequence is initially entered immediately following the execution of the restart sequence. The user is allowed to enter such pilot directed inputs as may be required through the request and data entry keyboards. The use of the RUN key terminates the data entry sequence.

The run sequence is, in essence, a continuous loop. The VOR and DME inputs are read and tested for validity. Slant-range correction is applied to the DME input when applicable. The position of the aircraft is computed and guidance information is displayed. The request latch is read, and the process repeated if no requests are made. The recognition of a request entry causes termination of the run sequence and entry into another data entry sequence. A RUN request returns program control to the run sequence.

A block diagram of the R-Nav System software is included in Appendix 6. For the sake of clarity, excessive detail has been omitted, but the reader should experience no difficulty in following the basic flow of control. Additional detail is provided in the software narrative description contained in Section 5.2.

5.1.3. Memory Utilization

The utilization of memory in the MAN System is, to some extent, determined by the architecture of the 6502. In order to take advantage of the short form address, page zero (Hex 00:FF) is used for all variables and the information pertaining to the waypoint currently in use. The variables which are displayed on the panel through the register file were chosen to occupy the addresses corresponding to the last two digits of the output address.

Page one, (Hex 100:1FF) is used for the stack, as mentioned previously. The addresses immediately above the stack (Hex 200:23B) are used for storing the information for ten waypoints. A small amount of RAM associated with the 6530 ROM chips (Hex 1780:17A7) is used for variables involved in Ground Speed computations. In any operational MAN System, this 599 bytes of memory would have to be implemented as RAM. For convenience, the CORDIC constants have also been stored in this area, but would be moved to ROM in an operational MAN System.

The restart sequence begins at hexadecimal address 023C, and the program and subroutines are stored between this location and 13F9. Generally, the main program is located in the lower portion, with the subroutines at the higher locations. Interrupt vectors are stored in locations 13FA through 13FF. The area between 023C and 13FF would be Read Only Memory in an operational MAN System.

5.2. Restart Sequence

The restart sequence is utilized to establish the machine configuration after a power on situation. The stack pointer is initialized at location 01FF, and the PIA data direction registers are established with PAD as output, and PBD as input. The active and way-point storage areas are cleared, and all displays are cleared. The request and status latches are also cleared.

The initialization of the real time clock is required near the end of the restart sequence. The numeric keyboard is read until four digits and a SET command have been entered. The validity of the input is tested to be between 0 and 23 hours, 59 minutes. The seconds value is automatically initialized at zero.

The keyboard entry routine is common to a number of other routines, and will be described at this point. Whenever keyboard input is required, the display device associated with the variable to be entered is cleared by writing the hexadecimal character "F" to all the displays in that group. The digit corresponding to the first digit to be entered is then overwritten with a hexadecimal "D" character. This character serves to prompt the user as to the digit to be entered. As digits are entered, this "entry indicator" is moved from left to right, and always indicates the next digit to be entered.

At any point in the entry routine, the clear entry (CE) key may be used to clear the preceding digit. Errors detected during entry may thus be corrected. After the appropriate number of digits is entered, the SET key must be pressed to terminate data entry.

All data entry routines check the validity of input values. An error condition is raised when an invalid entry is made, such as a bearing of 361.0 degrees. The error condition results in the calling of the FLASH4 routine which causes all four digits of the invalid data to flash on and off at the rate of approximately once a second. The flashing continues until the CE key is pressed. The return from the FLASH4 routine restarts the entire data entry routine for that input and forces the entry of correct data.

Once the setting for the real time clock has been entered, the restart sequence initializes the IRQ interrupt vector, enables interrupt requests, and starts the system clock. The clock is initialized by storing the value 244 in the interval timer. The system uses every 1024th clock cycle for counting. This results in generating an interrupt approximately every 249856 μsec. The seconds counter is incremented on every fourth interrupt. When the seconds counter reaches 60, it is reset to zero, and the minutes counter is incremented. When the minutes counter reaches a count of 60, the modulo 24 hours counter is incremented.

There is a software delay in the interrupt routine which is used to adjust the exact clock speed. The clock is intentionally set to run slightly fast to allow for the periods when the interrupt is disabled during DME routines.

Once the real time clock has been initialized, the restart sequence is complete. The service request sequence is entered directly from the restart sequence.

5.3. Service Request Entry

The service request sequence consists of a closed loop that is exited only when a RUN request is entered. In all other cases, the requests are serviced, the request latch cleared, and the loop repeated.

5.3.1. Omni Bearing Setting (OBS) Entry

The OBS entry routine utilizes the GETBRG routine for the entry of four digits. These digits are stored in the appropriate page zero address, and displayed on the output panel. Entries larger than 360.0 degrees are rejected. An entry of exactly 360.0 degrees is changed to 0.0 degrees.

5.3.2. Altitude Entry

The altitude entry utilizes a special case of a four digit entry. The high order digit is automatically set to zero, and is not displayed. The value entered by the user is the aircraft altitude above the station, in hundreds of feet. Entries larger than 199 are rejected.

5.3.3. Waypoint Entry

The waypoint entry routine is the most complex of the data entry routines. The first entry required is the waypoint number, which is a single digit between 0 and 9. Once this digit is entered, it is used as an offset to the base address of the waypoint storage area. The data associated with the entered waypoint number is moved from the waypoint storage area to the active (page zero) area. The information is simultaneously displayed on the panel.

At this point, two options are available to the user. He may press either the CE or the SET key. The SET key is intended for indicating that the waypoint just activated is to be used for navigation. Provided that the waypoint information has been previously initialized, depressing the SET key causes the request loop to be re-entered. A frequency entry of 0. is used to determine that the waypoint information has not been initialized. If the SET key is pressed when this condition exists, the frequency is flashed, and a valid entry for Frequency, Bearing, and Distance is forced.

After the waypoint information has been recalled and displayed, pressing the CE key will allow the entry, in sequence, of frequency, bearing, and distance. After all three are entered, the SET key is depressed to re-enter the request loop.

Waypoint data is checked for validity during entry. Bearings greater than 360.0 degrees or distances greater than 100.0 nautical miles are rejected. Frequency entries above 118.0 or below 108.0 are rejected. The odd frequencies between 108.0 and 112.0 are also rejected, since these frequencies are reserved for Instrument Landing System use.

5.3.4. Stop Entry

Depressing the STOP key effectively halts the R-Nav program. In an operational R-Nav System, it would result in entering a continuous loop, since the 6502 does not have a HALT state. This loop would only be terminated by a restart signal. In the MAN System, the STOP key effects a return to the KIM-1 monitor.

5.3.5. Test Entry

Facilities for testing the MAN System operation are provided by the TEST key. Depressing this key causes entry into a continuous loop which may only be terminated with the RUN key.

The test routine initially displays all 8's on all seven segment displays. All status indicators except Waypoint Alert are also illuminated. The operation of the system clock will continue, and will resume normal indications as seconds, minutes, and hours are updated.

The Distance to Waypoint, Bearing to Waypoint, and Waypoint Number will not display all 8's. The VOR bearing input will be displayed on the Bearing to Waypoint indicator. The DME distance will be displayed on the Distance to Waypoint indicator. A running count of the number of times the SEARCH routine (see below) is called is displayed on the Waypoint Number Indicator. The Waypoint Alert Indicator will be illuminated whenever the SEARCH routine is active and will be off when the track routine is being used.

This Test routine is intended primarily for ground use. It allows the testing of all outputs, and the verification of system inputs against known values while on the ground.

The test routine is terminated through the use of the RUN key. Returning from the routine is effected with all displays blank, except the system clock. The pilot directed input displays must be reactivated by using the normal entry routines. If the system is tested in flight, the computational displays will be reactivated whenever that particular data is re-computed and displayed. During an in-flight test, the actual data is not

lost, but the displays may be disabled for several minutes. In-flight testing of the system is not recommended. In fact, if the system is to be tested, this should be accomplished prior to any other data entry.

5.3.6. RUN Entry

The RUN entry effectively terminates the initial data entry sequence. The R-Nav computations begin as soon as the RUN key is depressed. This key should be depressed immediately before, or shortly after take-off. Valid radio inputs must be available for R-Nav computations to begin. If they are not available, the system will continue to attempt to obtain valid inputs, but no useful guidance information will be furnished by the system.

5.4. The RUN Sequence

There are two entry points in the RUN procedure. The first RUN command, and any RUN command executed after a change of VORTAC stations, utilizes the full RUN sequences. A shorter RUN sequence is used for looping purposes after the initialization steps in the full RUN sequence have been executed. The sequence which is described below includes the longer RUN sequence, with the loop re-entry point indicated.

5.4.1. RUN Initialization

The RUN sequence begins by clearing the request latch, and blanking all computational displays. The ALTCON subroutine is called to compute the altitude in nautical miles, rounded to the nearest 0.1 nmi. The CORDIC 1 (rotation mode) subroutine is called to compute the X and Y coordinates of the waypoint. These coordinates are stored in variables X2 and Y2 for later use.

The variable NC is set to 5 to indicate that a new six-minute ground speed computation is to begin. The variable NPS is set to zero to indicate the offset of the one minute interval X and Y coordinates which are used in the ground speed routine. The Ground Speed Not Established light is illuminated, and the current minute value is stored in a variable called NOW.

The DME SEARCH routine is called to establish the aircraft distance from the station. If this distance cannot be established after ten tries, the sequence is discontinued, and the Dead Reckoning (DEADR) sequence is entered. The initial bearing to the station is read and stored.

5.4.2. Main RUN Loop

At this point, the short loop begins. All looping for R-Nav computations returns to this point unless the station is changed.

The TRACK routine is called to find the latest distance from the station. If the TRACK routine fails to find a distance which is within $\pm$ 0.2 nmi. of the last distance, the SEARCH routine is called. If the SEARCH routine fails to obtain a distance, then the Dead Reckoning sequence (DEADR) is entered.

If the altitude in miles is not zero, CORDIC 3 is called to correct for the slant-range error, and, if the distance to the station is less than the altitude, the Station Passage Alert indicator is illuminated. If the altitude is zero, then the slant-range correction is not made, and the Station Passage Alert indicator is illuminated if the aircraft is within 2.0 nmi. of the station.

A new bearing is read and compared to the old bearing. If this bearing does not agree within one degree, the process is repeated ten times, with a 1/30th second time delay between tries. If no match can be established, the Dead Reckoning sequence is entered.

The Reciprocal of the bearing to the station is used to call CORDIC 1 to find the X and Y coordinates of this vector. These coordinates are added to the coordinates of the waypoint vector to form the coordinates of the vector from the aircraft to the waypoint. CORDIC 2 is called to convert these coordinates to the bearing and distance from the aircraft to the waypoint.

The COMP subroutine is used to compute the TO/FROM and LEFT/RIGHT relations, as well as the angle-off-track. CORDIC 1 is then used to compute distance-off-track. If the altitude is zero and the distance to the waypoint is less than 2.0 nmi., the Waypoint Alert indicator is illuminated. If the altitude is not zero, and the distance to the waypoint is less than or equal to the altitude, the Waypoint Alert indicator is illuminated.

All computed data is displayed on the front panel. The current minute is compared to the variable NOW. If these are not equal, NOW is updated to be equal to minute, and the ground speed update routine (SPEED) is called.

The request latch is read, and if no request has been made, the loop is repeated. If a request has been made, the request is serviced, and the loop repeated, unless the request involves a change of waypoint frequency. If a change of frequency is made, the loop is reinitiated from the first entry point. Since all coordinates are based on using the station as the origin, a change of station requires new computations.

5.5. The Special Algorithms

5.5.1. CORDIC Subroutine

The CORDIC algorithm is implemented as explained in Chapter 3. There are three options, chosen by a flag set by the calling program. CORDIC 1 is the rotation mode which performs polar to rectangular conversions, and is also used to compute distance-off-track. CORDIC 2 is the vectoring mode, used to perform rectangular to polar coordinate conversion. CORDIC 3 is the special form used to perform the slant-range correction for the DME distance.

5.5.2. DME SEARCH Subroutine

The SEARCH algorithm is used to establish an initial DME distance from the station. Interrupts are disabled during the period when responses to interrogation are being processed. The search begins by filling two separate arrays with all the responses received from two distinct interrogations. An array match routine then attempts to find two elements, one from each array, which agree within $\pm$0.2 nmi. If two such numbers are not found, the process is repeated up to ten times before a failure flag is returned. If matching numbers are found, one of these values is established as a test distance. The routine then uses this test distance as a basis for comparison for a number of new interrogations. When six responses in succession are found that agree within $\pm$0.2 nmi, the distance is established as equal to the test value, and control returns to the calling program. When five failures in succession are encountered, the SEARCH routine is reinitiated from the beginning. Ten failures of the entire algorithm results in returning a failure flag to the calling program.

5.5.3. DME TRACK Subroutine

The TRACK subroutine uses the established distance from the station as a basis for comparison with the returns from a single interrogation. If a match is found, the new distance is returned to the calling program. Ten failures to find a successful match results in the return of a failure flag to the calling program.

5.5.4. SPEED: Ground Speed Subroutine

The ground speed subroutine must be called six times, at one minute intervals, in order to establish the ground speed of the aircraft. The first five calls fill two five element circular queues with the X and Y coordinates of the aircraft at one minute intervals.

Once the queue is filled, and the routine is called for the sixth (or any subsequent) time, ground speed computations can take place. The oldest X and Y values in the queue are subtracted from the aircraft's current X and Y positions. These X and Y differences are used to call the CORDIC 2 subroutine, which finds the distance between the current and oldest position. This distance is divided by ten and rounded to give the ground speed in knots.

The six minute interval is used both to simplify the mathematics of the ground speed computation, and to stabilize any irregularities in individual positions. Once the ground speed is established, updating of the ground speed takes place at one minute intervals. A complete change of ground speed indication would take six minutes, but the trend would be noticable in one minute. This procedure is a compromise between the desire for rapid ground speed computations and the need to minimize the effects of small errors in position.

When the ground station is changed, the entire routine must be reinitiated from the beginning. Since the X and Y coordinates are computed with the station as the origin, coordinates computed based on one station would not be valid for use with another station. The station frequency is used to detect station changes.

5.5.5. DEADR: Dead Reckoning Sequence

When either of the inputs to the R-Nav System fails, or becomes unreliable, navigational computations become impossible. If this happens, the pilot must be informed of the situation and as much information presented to the pilot as possible.

In the event of the failure of an input prior to establishing the ground speed, little assistance can be given to the pilot. In this situation, the pilot is informed of the failure, and the display cleared. If one input remains reliable, this information is displayed to the pilot. If the DME is reliable, it is displayed on the Distance to Waypoint indicator. A reliable bearing is displayed on the Bearing to Waypoint indicator.

Continuous attempts are made to reacquire the lost signal. If the signal is regained, the system returns to the normal navigation mode. The request latch is also read in order to allow the user to select another station.

If failure occurs after the ground speed is established, the system can perform Dead Reckoning navigation based on the last ground speed. Using the oldest and newest X and Y values stored in the circular

queues, an X and Y increment for the last five minutes can be established. Dividing these values by 100 gives the X and Y changes for a three second interval.

Using the system clock, the current X and Y positions can be updated every three seconds to provide a reasonable approximation to the current position. The standard routines can then be used to compute the directional guidance information.

The Dead Reckoning capability is of limited usefulness. The positional computations are predicated on the aircraft maintaining the last known heading, altitude, and airspeed. Due to the lack of heading and airspeed inputs to the system, changes in these values would not be detected. This could result in the presentation of incorrect data to the pilot. If the pilot were to reverse course to return to the last known position, for example, the Dead Reckoning data would be completely useless.

The one practical use for the Dead Reckoning sequence would be when the aircraft is flying through the cone of confusion, directly over the station. Here the assumption of maintaining the last heading, altitude and airspeed would probably be justified, since the time in this cone seldom exceeds a few minutes.

An alternative to Dead Reckoning would be to have the pilot activate the test routine in flight whenever one input fails. This will provide the operational guidance information directly to the pilot, as described in Section 5.3.5.

In any event, the approach of taking the input information from the early stages of both radios leaves both VOR and DME radios in their original form. In the event of failure of one radio, the other would still be available for guidance completely independent of the R-Nav System.

Due to memory restrictions, the Dead Reckoning algorithm was not implemented in the MAN System.

6. RESULTS AND CONCLUSIONS

6.1. R-Nav Results

The MAN System was constructed and programmed as outlined in the preceding chapters. The system has been tested under static conditions with excellent results.

The entire system was constructed at a total hardware cost of less than $750, excluding power supply. This is considering the fact that most components were purchased in quantities of one, the most expensive way to buy electronic components. A very tentative estimate is that the system could be built in quantity for a price of well below $500, excluding radios and displays. Even allowing for a 300 percent mark-up, this represents an end user cost of about $2,000. This is considerably below the average cost quoted in Chapter 1, and, with a single exception, it is substantially below the cost of any currently available unit.

A short system test patch was added to the RUN routine to count the number of computations per minute. This figure will vary somewhat depending on two factors. If the altitude is not zero, the slant-range correction must be made during each iteration of the main loop. This requires approximately 40 milliseconds. If the altitude is zero, this correction is not required, and hence, a considerable amount of time is saved. Changing the altitude from zero to some other value results in the loss of about 90 computations per minute.

The algorithm speed is to some extent position dependent. At several points in the algorithm, negative numbers must be changed to positive, computations performed, and the results changed back to negative representation. The exact number of times this must be performed will also determine the speed of the algorithm.

With the inputs static, that is, with few extra DME searches, and no extra Bearing Reads required, the algorithm is considerably faster than is required. In general, the algorithm runs between 300 and 500 computations per minute. This is between five and eight computations per second. In the worst case, this represents a positional error of less than 0.013 nautical miles. This is well within the limitations specified in Chapters 2 and 3.

If the time for computation is treated as a time error, the maximum aircraft speed which can be handled by the system is readily computed. The maximum allowable time for computation error was computed in Chapter 3 to be 0.4865 nautical miles. If a pessimistic rate of five computations per second is considered, then the maximum speed would be:

$$\text{RATE} = \text{DIST}/\text{TIME}$$

or

$$\text{RATE} = (0.4865/.2)\cdot 3600$$

This gives a speed of about 8,750 nautical miles per hour. This is considerably higher than the 240 knot maximum speed assumed at the beginning of the research.

The DME system was also tested separately and performs a very close approximation to the actual ground station. With the station disabled, an occasional lock-on to a nonexistant station was observed. These were quickly discarded by the system. True DME lock-ons were occasionally lost, but the system quickly recovered without interruption of computations.

6.2. Conclusions

It can be safely stated that a microprocessor, such as the 6502, is quite capable of performing R-Nav computations and producing results within FAA tolerances. Such a system would require slightly more than 4K of Read-Only Memory, and less than 1K of Random Access Memory.

More importantly, the results have shown that a relatively slow microprocessor, with limited arithmetic capabilities can be adapted to meet rather stringent operational requirements. The problems encountered in this design are common to the more general question of adapting microprocessors to the solution of a variety of real-time applications. These results indicate that a dedicated microprocessor, supported by a small amount of specialized interface hardware and a well-designed software support system, can offer a viable alternative to special purpose discrete logic systems. The low cost of microprocessors can, as in the case of the MAN System, provide a significant economic advantage to systems of this type.

Such applications may often require very specialized computation techniques, such as the use of the CORDIC algorithm in the MAN System. Specialized interfaces may also be required, as in the case of the DME input to the MAN System. These specialized techniques, however, represent only minor variations in the basic structure of such a system. These are the features which adapt the microprocessor to the specific environment in which it must operate.

The common feature of all such designs will be the microprocessor, around which other system components are assembled. The basic adaptability of the microprocessor, when combined with the flexibility inherent in any software-driven system, will provide a very generalized solution to a large variety of specialized problems.

6.3. Suggested Expansion and Further Research

The MAN System has proven the feasibility of a microprocessor-based R-Nav System. The system has, to date, however, only been tested under static conditions. That is, the inputs can only be varied by changing the switch settings on the front panel. Since it is impossible to test all possible waypoint and aircraft positions in this manner, dynamic testing, to include the full range of possible inputs, would be a logical next step. Such testing would require a source of phase-varying sine waves to simulate the VOR inputs. A fairly complex system for providing the DME signals would also be required. Lacking these facilities, another microprocessor could be programmed to simulate an entire flight, and these inputs fed directly to the MAN System.

Alternative methods of acquiring the DME responses should also be investigated. The MAN System relies on the DME radio to recognize the unique pulse-pair configuration which represents a response. Some initial investigation revealed that the 6502 is not fast enough to detect the "101" configuration of this pulse directly from the receiver. A 2 MHz version of the 6502 is now available, which may be fast enough to perform this task. This would represent a slight increase in system cost, but might be justified in view of the added redundancy provided in the system.

There is a distinct possibility that specialized radios could, and should, be provided as a component of the MAN System. These radios would be considerably less expensive than current radios, since none of the analog computing circuits would be required. In this manner, an aircraft already equipped with VOR and DME radios would have two completely independent systems for navigation.

Another area for exploration would be the use of two VORTAC radios, both of which provide data to the R-Nav System. Such a system would have excellent back-up capability in the event one radio fails. Further, this system could use a weighted sum of inputs to obtain results which are more accurate than those relying on a single radio source. A more complex algorithm would be required to determine which of the signals was more reliable, but might be well worth the effort.

The MAN System uses raw data from the inputs. Large momentary fluctuations of these inputs, as often happens with VOR scalloping, can result in erroneous results. Since these are of short duration, they can be accepted but filtering and smoothing techniques for both inputs would be worth investigating.

Finally, the ultimate extension of the MAN System would be installation and flight testing. After all other tests are completed, this would be the final proof of the applicability of microprocessors to provide R-Nav capabilities for general aviation.

LIST OF REFERENCES

[1] Federal Aviation Administration, Office of Aviation Economics, Aviation Forecast Division. Aviation Forecasts: Fiscal Years 1970-1981, January 1970, pp. 12.

[2] Federal Aviation Administration. Application of Area Navigation in the National Airspace System, a report of the FAA/Industry Task Force, February 1973.

[3] Federal Aviation Administration. The National Aviation System Plan, 1973-1982, March 1973, pp. 90.

[4] Mancill, Julian D., Modern Analytical Trigonometry, Dood, Mead and Company, Inc., 1960, pp. 219.

[5] United States Air Force, AF Manual 51-37, Instrument Flying, July 22, 1968, pp. 12-4.

[6] Airplane Owners and Pilots Association, "Avionics Bonus Supplement," The AOPA Pilot, June 1975, pp. 82.

[7] Federal Aviation Administration. Flight Standards Training Branch, Rho-Theta Navigation Systems, Manual No. FI-202/203B, 1969, pp. 2-1.

[8] Selby, Samuel M., editor, Standard Mathematical Tables, 15th edition, Chemical Rubber Co., 1967, pp. 11.

[9] Federal Aviation Administration. Advisory Circular AC 90-45: Approval of Area Navigation Systems for Use in the U.S. National Airspace System, 18 August 1969.

[10] Bowker, Albert H. and Lieberman, Gerald J., Engineering Statistics, Prentice-Hall, Inc., 1959, pp. 555.

[11] Jeppesen and Company, Airway Manual, Radio Facilities Section, May 1969, pp. 15.

[12] Kendall, Maurice George and Moran, P. A. P., Geometrical Probability, Hafner Publishing Co., 1963, pp. 66.

[13] Volder, Jack E., "CORDIC Trigonometric Computing Technique," IRE Transactions on Electronic Computers, EC-8, September 1959, pp. 330.

[14] Schmid, Hermann, Decimal Computation, John Wiley and Sons, 1974, pp. 162.

[15] Schmid, Hermann and Bogacki, Anthony, "Use Decimal CORDIC For Generation of Many Transcendental Functions," Electronic Data News, February 20, 1973, pp. 64.

[16] Cornell University, Users Guide to PL/C The Cornell Compiler for PL/I, Release 7, June 1964, pp. 19.

[17] MOS Technology, MCS6500 Microcomputer Family Hardware Manual, Publication No. 6500-10, August 1975.

[18] MOS Technology, MCS6500 Microcomputer Family Programming Manual, Publication No. 6500-50, August 1975.

[19] MOS Technology, KIM-1 Microcomputer Module User Manual, Publication No. 6500-15, January 1976.

[20] Lancaster, Don, TTL Cookbook, Howard W. Sams and Company, Inc., 1975, pp. 281.

APPENDIX 1

PL/C Simulation of CORDIC Area Navigation Computations

```
STATS:PROCEDURE OPTIONS(MAIN);
 /*                                                    */
 /* PROGRAM WHICH IS DESIGNED TO SIMULATE THE          */
 /* CALCULATIONS PERFORMED BY THE CORDIC               */
 /* ALGORITHM AS USED IN AN AREA NAVIGATION            */
 /* SYSTEM.  RANDOW WAYPOINTS AND ALTITUDES ARE        */
 /* ARE CHOSEN BY MONTE-CARLO TECHNIQUES FOR           */
 /* EACH OF 121 AIRCRAFT POSITIONS.  DOUBLE            */
 /* PRECUSION RESULTS FOR THE SAME CALCULATIONS        */
 /* ARE USED AS AS A BASIS OF COMPARISON FOR           */
 /* COLLECTING STATISTICS ABOUT THE ERRORS             */
 /* INVOLVED IN THE CORDIC COMPUTATIONS.               */
 /*                                                    */
      DCL (ALGTRK,CRSTRK) DEC FLOAT(15);
      DCL ( XDIST(-100:100),YDIST(-100:100),TDIST(-100:100))
      DEC FIXED(4);
      DCL(TOTXERR,TOTYERR,TOTERR,TOTXSQ,TOTYSQ,TOTSQ,SIGX,SIGY,SIG,
      XAVG,YAVG,AVG,MAXX,MAXY,MAX) DEC FLOAT(15) INIT (0.0);
      DCL(R,TH,AF,AM,HR,X,Y,CXR,CYR,ERR) DEC FLOAT(15);
      DCL (RW,THW,HRW,XERR,YERR,XW,YW) DEC FLOAT (15);
      DCL(CR1,CR2,CTH1,CTH2,CAF,CRWB,TDUM,CAM,CRH1,CRH2,
      CRR,CTHR) DEC FIXED(4,1);
      DCL (X1,Y1,X2,Y2,CX,CY ) DEC FIXED(8,5);
      DCL T1 DEC FIXED(5,0);
     DCL (ALPHA(4),C) DEC FIXED   (8,5);
     DCL Z DEC FLOAT(9);
      DCL(XREG,YREG,ANG,XTEMP,YTEMP) DEC FIXED(8,5);
CORDIC1:PROCEDURE(R,THETA,X,Y);
 /*                                                    */
 /* CORDIC 1 IS THE ROTATION MODE OF THE CORDIC        */
 /* ALGORITHM.  IT IS USED AS A MEANS OF               */
 /* PERFORMING POLAR TO RECTANGULAR COORDINATE         */
 /* CONVERSION.                                        */
 /*                                                    */
      DCL (R,THETA) DEC FIXED (4,1);
      DCL(X,Y) DEC FIXED(8,5);
       ANG=THETA;
       IF ANG>180 THEN ANG=ANG-360;
       XREG=R;
       YREG=0;
       IF ANG>0 THEN E=-1; ELSE E=1;
       ANG=ANG+E*90;
       XTEMP=E*YREG;
       YREG=-E*XREG;
       XREG=XTEMP;
       IF ANG>0 THEN E=-1; ELSE E=1;
       ANG=ANG+E*45;
```

```
        XTEMP=XREG+E*YREG;
        YREG=YREG-E*XREG;
        XREG=XTEMP;
        DO I=1 TO 4;
          DO J=1 TO 9;
            IF ANG>0 THEN E=-1; ELSE E=1;
            XTEMP=XREG+E*(YREG/(10**I));
            YREG=YREG-E*(XREG/(10**I));
            XREG=XTEMP;
            ANG=ANG+E*ALPHA(I);
          END;
        END;
      X=DIVIDE(XREG,C, 8,5);
      Y=DIVIDE(YREG,C, 8,5);
END CORDIC1;
CORDIC2:PROCEDURE(R,THETA,X,Y);
 /*                                                    */
 /* CORDIC 2 IS THE VECTORING MODE OF THE              */
 /* CORDIC ALGORITHM.  IT IS USED AS A MEANS OF        */
 /* RECTANGULAR TO POLAR COORDINATE CONVERSION.        */
 /*                                                    */
       DCL (R,THETA) DEC FIXED (4,1);
       DCL(X,Y) DEC FIXED(8,5);
         XREG=X;
         YREG=Y;
         ANG=0;
         IF XREG*YREG<0 THEN E=-1; ELSE E=1;
         ANG=ANG+E*90;
         XTEMP=E*YREG;
         YREG=-E*XREG;
         XREG=XTEMP;
         IF XREG*YREG<0 THEN E=-1; ELSE E=1;
         ANG=ANG+E*45;
         XTEMP=XREG+E*YREG;
         YREG=YREG-E*XREG;
         XREG=XTEMP;
         DO I=1 TO 4;
           DO J=1 TO 9;
             IF XREG*YREG<0 THEN E=-1; ELSE E=1;
             ANG=ANG+E*ALPHA(I);
             XTEMP=XREG+E*(YREG/(10**I));
             YREG=YREG-E*(XREG/(10**I));
             XREG=XTEMP;
           END;
         END;
         R=DIVIDE(XREG,C,8,5)+.05;
         IF R<=0 THEN DO;
           R=-R;
           ANG=ANG+180;
         END;
         IF ANG<0 THEN ANG=ANG+360;
         IF ANG>=360 THEN ANG=ANG-360;
         THETA=ANG+.05;
END CORDIC2;
CORDIC3:PROCEDURE(R,THETA,X,Y);
```

```
 /*                                                    */
 /* CORDIC 3 IS A SPECIAL FORM OF THE CORDIC           */
 /* ALGORITHM WHICH IS USED TO PERFORM THE             */
 /* SLANT-RANGE CORRECTION FOR THE DME DISTANCE.       */
 /*                                                    */
      DCL (R,THETA,X,Y) DEC FIXED(4,1);
        XREG=R;
        YREG=0;
        ANG=0;
        YTEMP=MULTIPLY(Y,C,8,5);
        IF YTEMP>YREG&XREG<0^YTEMP<YREG&XREG>0 THEN E=1; ELSE E=-1;
        ANG=ANG+E*45;
        XTEMP=XREG+E*YREG;
        YREG=YREG-E*XREG;
        XREG=XTEMP;
        DO I= 1 TO 4;
          DO J=1 TO 9;
            IF YTEMP>YREG&XREG<0^YTEMP<YREG&XREG>0 THEN E=1; ELSE E=-1;
            ANG=ANG+E*ALPHA(I);
            XTEMP=XREG+E*(YREG/(10**I));
            YREG=YREG-E*(XREG/(10**I));
            XREG=XTEMP;
          END;
        END;
        X=DIVIDE(XREG,C,8,5)+.05;
        THETA=ANG+.05;
END CORDIC3;
 /*                                                    */
 /* SET PROGRAM CONSTANTS                              */
 /*                                                    */
     ALPHA(1)=5.71059;
     ALPHA(2)=0.57294;
     ALPHA(3)=0.05730;
     ALPHA(4)=0.00573;
     C=1.47965;
      Z=0.391477675;
 /*                                                    */
 /* MAIN PROGRAM LOOP                                  */
 /*                                                    */
     DO ALGTRK=0 TO 100 BY 10;
      DO CRSTRK=0 TO 100 BY 10;
      IF ALGTRK=0 THEN TH=90;ELSE TH=ATAND(CRSTRK/ALGTRK);
      CTH2=FIXED(TH+.05,4,1);
      HR=SQRT(ALGTRK*ALGTRK+CRSTRK*CRSTRK);
      XDIST,YDIST,TDIST=0;
      TOTERR,TOTXERR,TOTYERR=0;
      TOTXSQ,TOTYSQ,TOTSQ=0;
      SIGX,SIGY,SIG=0;
      XAVG,YAVG,AVG=0;
      MAXX,MAXY,MAX=0;
 /*                                                    */
 /* DO 1000 RANDOM WAYPOINTS AND ALTITUDES.            */
 /*                                                    */
      DO KKK=1 TO 1000;
 /*                                                    */
```

```
/* PICK A RANDOM DISTANCE                                    */
/*                                                           */
      Z=RAND(Z);
      DO WHILE(Z>0.1);
        Z=RAND(Z);
      END;
      HRW=Z*1000;
      CR1=FIXED(HRW+.05,4,1);
/*                                                           */
/* PICK A RANDOM BEARING                                     */
/*                                                           */
      Z=RAND(Z);
      DO WHILE(Z>0.36);
        Z=RAND(Z);
      END;
      THW=Z*1000;
      CTH1=FIXED(THW+0.05,4,1);
/*                                                           */
/* PICK A RANDOM ALTITUDE                                    */
/*                                                           */
      Z=RAND(Z);
      DO WHILE(Z>0.199);
        Z=RAND(Z);
      END;
      AF=Z*100000;
      CAF=FIXED(Z*1000+.5,3,0);
      AM=AF/FLOAT(6076.10333333333,15);
/*                                                           */
/*     REFERENCE CALCULATIONS                                */
/*                                                           */
      R=SQRT(HR*HR+AM*AM);
      CR2=FIXED(R+0.05,4,1);
      X=HR*COSD(TH);
      Y=HR*SIND(TH);
      XW=HRW*COSD(THW);
      YW=HRW*SIND(THW);
/*                                                           */
/*     CORDIC CALCULATIONS                                   */
/*                                                           */
      CRWB=CTH1-180.0;
      IF CRWB<0 THEN CRWB=CRWB+360.0;
      T1=CAF*100;
      CAM=MULTIPLY(T1,0.0001646,5,2)+0.05;
      CALL CORDIC3(CR2,TDUM,CRH2,CAM);
      CALL CORDIC1(CR1 ,CRWB,X1,Y1);
      CALL CORDIC1(CRH2,CTH2,X2,Y2);
      CX=X1+X2;
      CY=Y1+Y2;
      CALL CORDIC2(CRR,CTHR,CX,CY);
/*                                                           */
/*     COMPARISON                                            */
/*                                                           */
      CXR=FLOAT(CRR,15)*COSD(FLOAT(CTHR,15))+XW;
      CYR=FLOAT(CRR,15)*SIND(FLOAT(CTHR,15))+YW;
      XERR=X-CXR;
```

```
      YERR=Y-CYR;
      ERR=SQRT((X-CXR)**2+(Y-CYR)**2);
/*                                                    */
/* COLLECT STATISTICS ON ERRORS.                      */
/*                                                    */
      IF XERR<-10 THEN XDIST(-100)=XDIST(-100)+1;
        ELSE IF XERR>10  THEN XDIST(100)=XDIST(100)+1;
          ELSE DO;
            SCPT=XERR*100;
            IF SCPT>0 THEN SCPT=SCPT+.5; ELSE SCPT=SCPT-.5;
            XDIST(SCPT)=XDIST(SCPT)+1;
          END;
      IF YERR<-10 THEN YDIST(-100)=YDIST(-100)+1;
        ELSE IF YERR>10  THEN YDIST(100)=YDIST(100)+1;
          ELSE DO;
            SCPT=YERR*100;
            IF SCPT>0 THEN SCPT=SCPT+.5; ELSE SCPT=SCPT-.5;
            YDIST(SCPT)=YDIST(SCPT)+1;
          END;
      IF  ERR<-10 THEN TDIST(-100)=TDIST(-100)+1;
        ELSE IF  ERR>10  THEN TDIST(100)=TDIST(100)+1;
          ELSE DO;
            SCPT= ERR*100;
            IF SCPT>0 THEN SCPT=SCPT+.5; ELSE SCPT=SCPT-.5;
            TDIST(SCPT)=TDIST(SCPT)+1;
          END;
      IF ABS(XERR)>ABS(MAXX) THEN MAXX=XERR;
      IF ABS(YERR)>ABS(MAXY) THEN MAXY=YERR;
      IF ERR>MAX THEN MAX=ERR;
      TOTXERR=TOTXERR+XERR;
      TOTYERR=TOTYERR+YERR;
      TOTERR=TOTERR+ERR;
      TOTXSQ=TOTXSQ+XERR*XERR;
      TOTYSQ=TOTYSQ+YERR*YERR;
      TOTSQ=TOTSQ+ERR*ERR;
    END;
    N=KKK-1;
    XAVG=TOTXERR/N;
    YAVG=TOTYERR/N;
    AVG=TOTERR/N;
    SIGX=TOTXSQ/N-XAVG*XAVG;
    SIGY=TOTYSQ/N-YAVG*YAVG;
    SIG=TOTSQ/N-AVG*AVG;
```

```
/*                                                    */
/* PRINT STATISTICS ON 1000 RANDOM WAYPOINTS          */
/* FOR ONE AIRCRAFT POSITION                          /*
/*                                                    */
     PUT SKIP (3) EDIT(ALGTRK,CRSTRK)(F(3),X(3),F(3));
     PUT SKIP    LIST('X DATA',XAVG,SIGX,MAXX);
     PUT SKIP LIST('Y DATA',YAVG,SIGY,MAXY);
     PUT SKIP LIST('TOTAL DATA',AVG,SIG,MAX);
     DO JJ=-100 TO 100;
       PUT SKIP LIST(JJ,XDIST(JJ),YDIST(JJ),TDIST(JJ));
     END;
     END;

     END;
 END STATS;
```

APPENDIX 2

Summary of Distribution of Errors in CORDIC Area Navigation Computations

APPENDIX 2
POSITIONAL ERROR OF CORDIC ROUTINE

		X= 0		X= 10		X= 20	
		MEAN	VARIANCE	MEAN	VARIANCE	MEAN	VARIANCE
Y= 0	X	-0.0313	0.00223	-0.0285	0.00305	0.0329	0.00317
	Y	-0.0140	0.01157	0.0081	0.00458	0.0157	0.00446
Y= 10	X	-0.0261	0.00351	-0.0196	0.00427	-0.0441	0.00336
	Y	-0.0018	0.00356	0.0069	0.00509	0.0032	0.00549
Y= 20	X	-0.0190	0.00291	-0.0210	0.00427	-0.0226	0.00351
	Y	0.0011	0.00333	0.0128	0.00358	0.0048	0.00406
Y= 30	X	-0.0136	0.00288	-0.0155	0.00382	-0.0291	0.00384
	Y	0.0024	0.00307	0.0214	0.00311	-0.0149	0.00388
Y= 40	X	-0.0100	0.00256	-0.0111	0.00418	0.0085	0.00365
	Y	0.0048	0.00288	0.0311	0.00313	-0.0165	0.00337
Y= 50	X	-0.0072	0.00234	-0.0104	0.00276	-0.0260	0.00411
	Y	0.0026	0.00285	0.0132	0.00270	-0.0047	0.00328
Y= 60	X	-0.0034	0.00203	0.0061	0.00394	-0.0259	0.00441
	Y	0.0022	0.00304	-0.0360	0.00294	0.0372	0.00315
Y= 70	X	-0.0029	0.00198	-0.0032	0.00287	-0.0162	0.00212
	Y	0.0024	0.00337	0.0370	0.00305	0.0548	0.00319
Y= 80	X	-0.0039	0.00213	-0.0031	0.00371	-0.0385	0.00318
	Y	0.0025	0.00328	0.0288	0.00311	0.0409	0.00315
Y= 90	X	-0.0009	0.00207	-0.0452	0.00281	-0.0137	0.00255
	Y	0.0038	0.00363	0.0514	0.00366	0.0476	0.00348
Y=100	X	0.0013	0.00197	-0.0018	0.00199	-0.0223	0.00228
	Y	-0.0024	0.00385	0.0238	0.00410	0.0210	0.00392

APPENDIX 2
POSITIONAL ERROR OF CORDIC ROUTINE

		X= 30		X= 40		X= 50	
		MEAN	VARIANCE	MEAN	VARIANCE	MEAN	VARIANCE
Y= 0	X	-0.0254	0.00315	-0.0264	0.00291	0.0238	0.00351
	Y	0.0204	0.00485	0.0222	0.00428	0.0298	0.00410
Y= 10	X	-0.0001	0.00310	0.0070	0.00310	-0.0062	0.00340
	Y	0.0101	0.00528	0.0132	0.00602	0.0147	0.00450
Y= 20	X	-0.0351	0.00326	-0.0392	0.00290	-0.0239	0.00339
	Y	-0.0023	0.00447	0.0298	0.00458	-0.0037	0.00550
Y= 30	X	-0.0062	0.00341	-0.0319	0.00262	-0.0431	0.00319
	Y	0.0118	0.00429	0.0264	0.00349	0.0305	0.00342
Y= 40	X	0.0091	0.00251	-0.0139	0.00350	-0.0432	0.00366
	Y	-0.0151	0.00310	0.0000	0.00412	0.0380	0.00422
Y= 50	X	0.0187	0.00279	0.0185	0.00375	-0.0017	0.00301
	Y	-0.0260	0.00327	-0.0269	0.00419	0.0055	0.00338
Y= 60	X	0.0136	0.00350	0.0062	0.00289	-0.0079	0.00269
	Y	-0.0401	0.00332	-0.0072	0.00352	0.0151	0.00322
Y= 70	X	-0.0259	0.00328	-0.0375	0.00365	-0.0327	0.00381
	Y	-0.0083	0.00344	0.0632	0.00380	0.0453	0.00379
Y= 80	X	0.0124	0.00415	0.0135	0.00416	-0.0155	0.00440
	Y	-0.0603	0.00341	-0.0569	0.00388	0.0048	0.00433
Y= 90	X	-0.0420	0.00273	0.0181	0.00255	0.0010	0.00251
	Y	0.0479	0.00395	-0.0648	0.00370	-0.0921	0.00373
Y=100	X	0.0023	0.00237	0.0034	0.00231	0.0331	0.00262
	Y	-0.0095	0.00414	-0.0020	0.00406	-0.0639	0.00417

APPENDIX 2
POSITIONAL ERROR OF CORDIC ROUTINE

		X= 60		X= 70		X= 80	
		MEAN	VARIANCE	MEAN	VARIANCE	MEAN	VARIANCE
Y= 0	X	-0.0248	0.00340	-0.0194	0.00372	-0.0199	0.00387
	Y	0.0348	0.00404	0.0413	0.00421	0.0433	0.00389
Y= 10	X	-0.0590	0.00318	0.0172	0.00356	0.0134	0.00383
	Y	0.0349	0.00557	0.0275	0.00454	0.0289	0.00572
Y= 20	X	0.0190	0.00338	0.0433	0.00345	0.0319	0.00371
	Y	-0.0024	0.00570	0.0092	0.00376	-0.0086	0.00459
Y= 30	X	-0.0500	0.00316	-0.0179	0.00329	-0.0752	0.00354
	Y	0.0289	0.00393	-0.0064	0.00441	0.0351	0.00595
Y= 40	X	-0.0186	0.00318	0.0546	0.00349	-0.0654	0.00377
	Y	0.0251	0.00369	-0.0154	0.00444	0.0322	0.00508
Y= 50	X	-0.0006	0.00274	0.0395	0.00355	-0.0043	0.00396
	Y	0.0076	0.00319	-0.0200	0.00464	-0.0025	0.00533
Y= 60	X	-0.0276	0.00316	-0.0026	0.00336	-0.0445	0.00322
	Y	-0.0215	0.00351	0.0079	0.00330	0.0440	0.00361
Y= 70	X	-0.0027	0.00271	-0.0008	0.00318	-0.0152	0.00330
	Y	0.0052	0.00339	0.0027	0.00377	0.0215	0.00361
Y= 80	X	0.0306	0.00266	0.0130	0.00320	-0.0070	0.00440
	Y	-0.0390	0.00340	-0.0109	0.00365	-0.0048	0.00472
Y= 90	X	-0.0132	0.00266	0.0293	0.00380	-0.0464	0.00385
	Y	-0.0367	0.00376	-0.0295	0.00449	0.0597	0.00425
Y=100	X	0.0434	0.00348	-0.0189	0.00300	0.0267	0.00331
	Y	-0.0684	0.00444	-0.0291	0.00412	-0.0917	0.00410

APPENDIX 2
POSITIONAL ERROR OF CORDIC ROUTINE

		X= 90 MEAN	X= 90 VARIANCE	X=100 MEAN	X=100 VARIANCE
Y= 0	X	-0.0201	0.00407	-0.0188	0.00460
	Y	0.0460	0.00437	0.0466	0.00401
Y= 10	X	0.0406	0.00384	0.0125	0.00452
	Y	-0.0078	0.00482	0.0316	0.00437
Y= 20	X	0.0371	0.00399	0.0110	0.00444
	Y	0.0172	0.00411	0.0080	0.00385
Y= 30	X	0.0380	0.00381	-0.0146	0.00462
	Y	-0.0177	0.00425	0.0219	0.00346
Y= 40	X	-0.0726	0.00395	-0.0084	0.00422
	Y	0.0391	0.00367	0.0199	0.00374
Y= 50	X	-0.0956	0.00357	-0.0662	0.00431
	Y	0.0166	0.00330	0.0457	0.00348
Y= 60	X	-0.0388	0.00398	-0.0666	0.00449
	Y	-0.0027	0.00350	0.0479	0.00450
Y= 70	X	-0.0340	0.00420	-0.0326	0.00459
	Y	0.0351	0.00437	-0.0142	0.00342
Y= 80	X	0.0576	0.00391	-0.0939	0.00403
	Y	-0.0445	0.00407	0.0298	0.00361
Y= 90	X	-0.0078	0.00394	-0.0241	0.00538
	Y	-0.0173	0.00392	0.0172	0.00523
Y=100	X	0.0155	0.00516	0.0106	0.00478
	Y	-0.0218	0.00510	0.0016	0.00493

MEAN OF X ERRORS	-1.06392131778126E-02
STANDARD DEVIATION OF X ERRORS	1.46353911890315E-03
MEAN OF Y ERRORS	6.13398962801150E-03
STANDARD DEVIATION OF Y ERRORS	2.09683498321473E-03

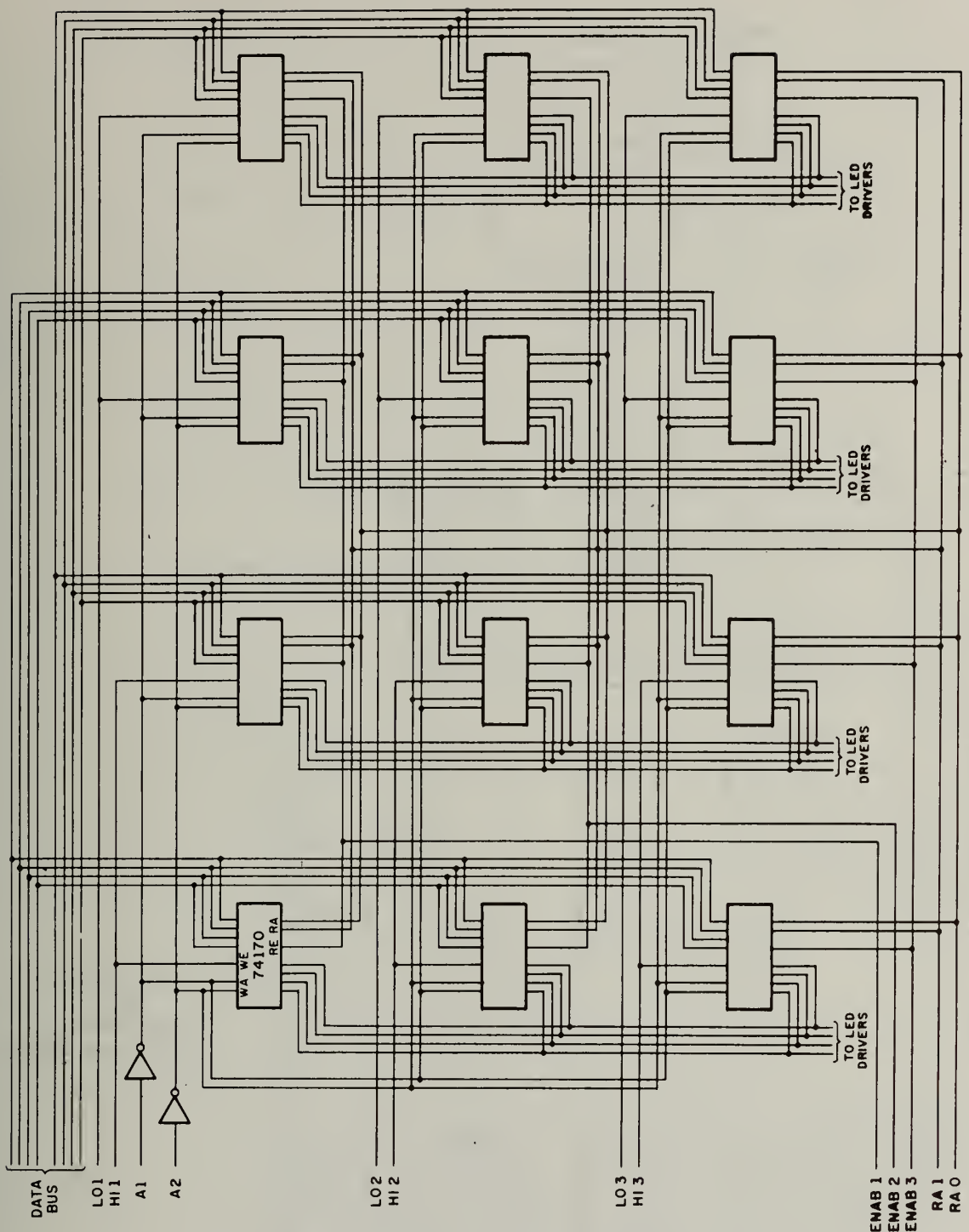
APPENDIX 3

Summary of Least Squares Analysis of Errors

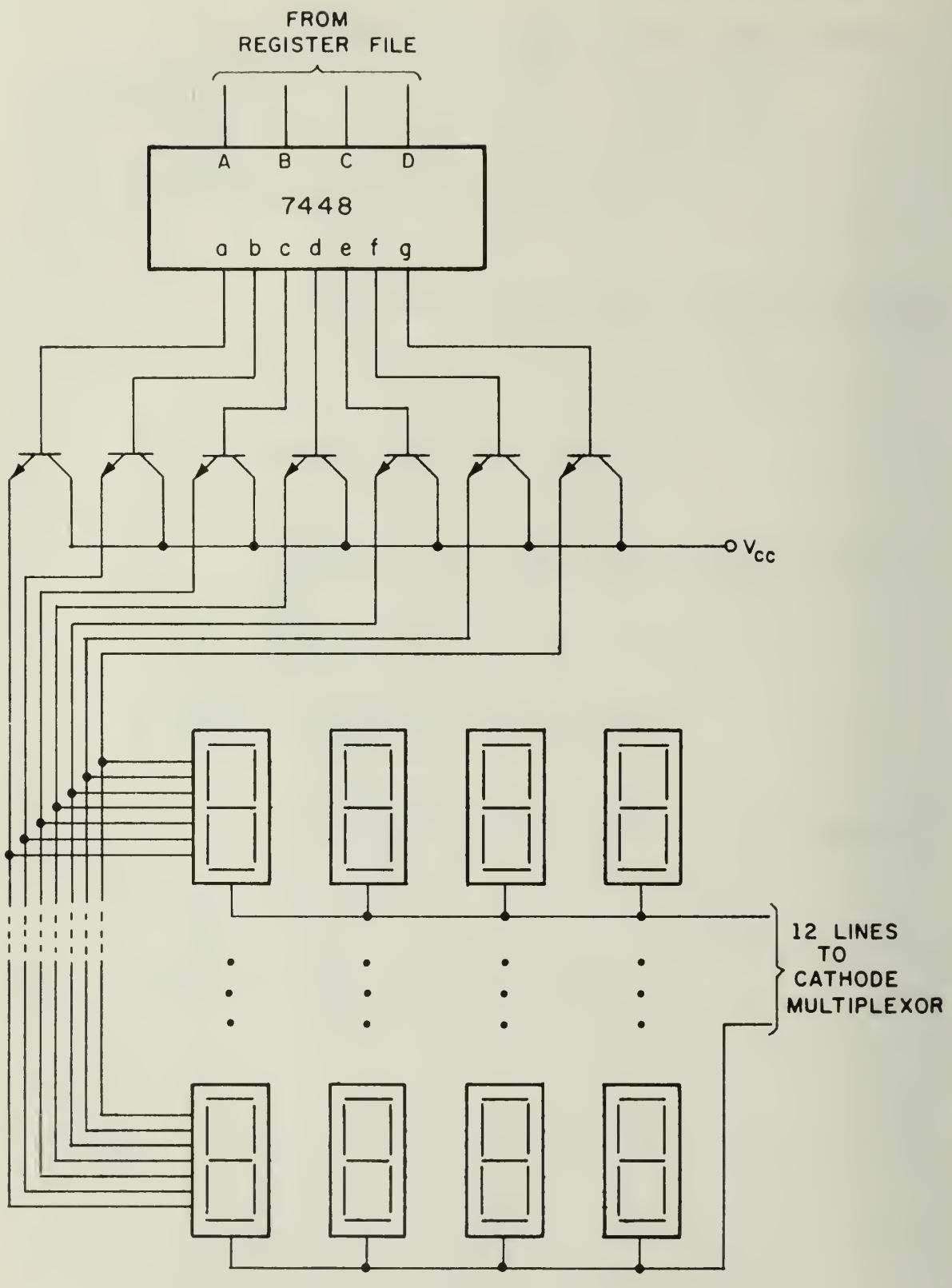
Variable	X	Y
Sample Mean	-0.3290849D-01	0.2946550D-01
Sample Standard Deviation	0.5256766D-01	0.5684478D-01
Sample Size	1000	1000
CHI-SQUARE Value	0.1008000D 03	0.9980000D 02
Degrees of Freedom	97	97
Probability Greater than CHI-SQUARE	0.3755937D 00	0.4024930D 00

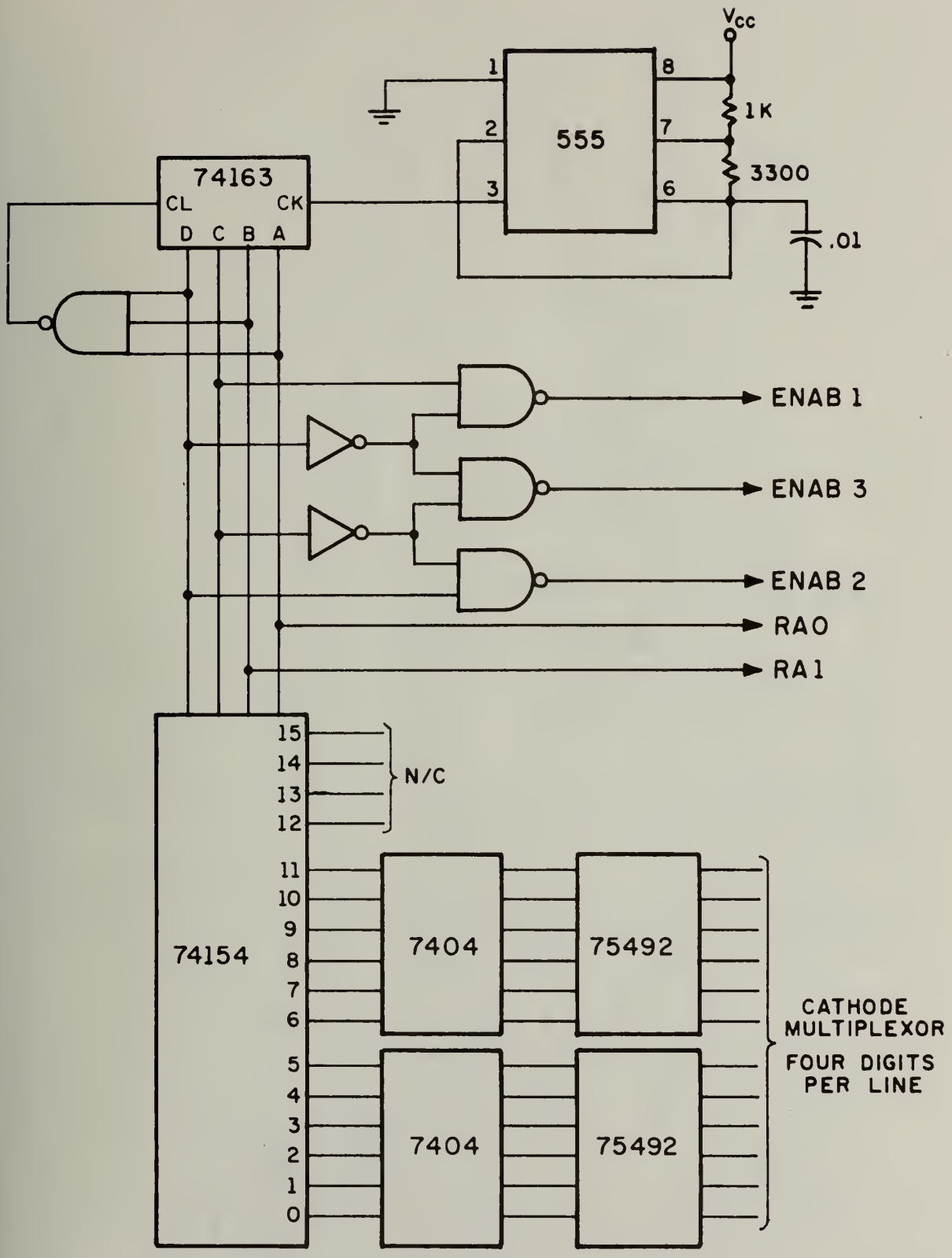
APPENDIX 4

Register File and Display Multiplexor Circuits



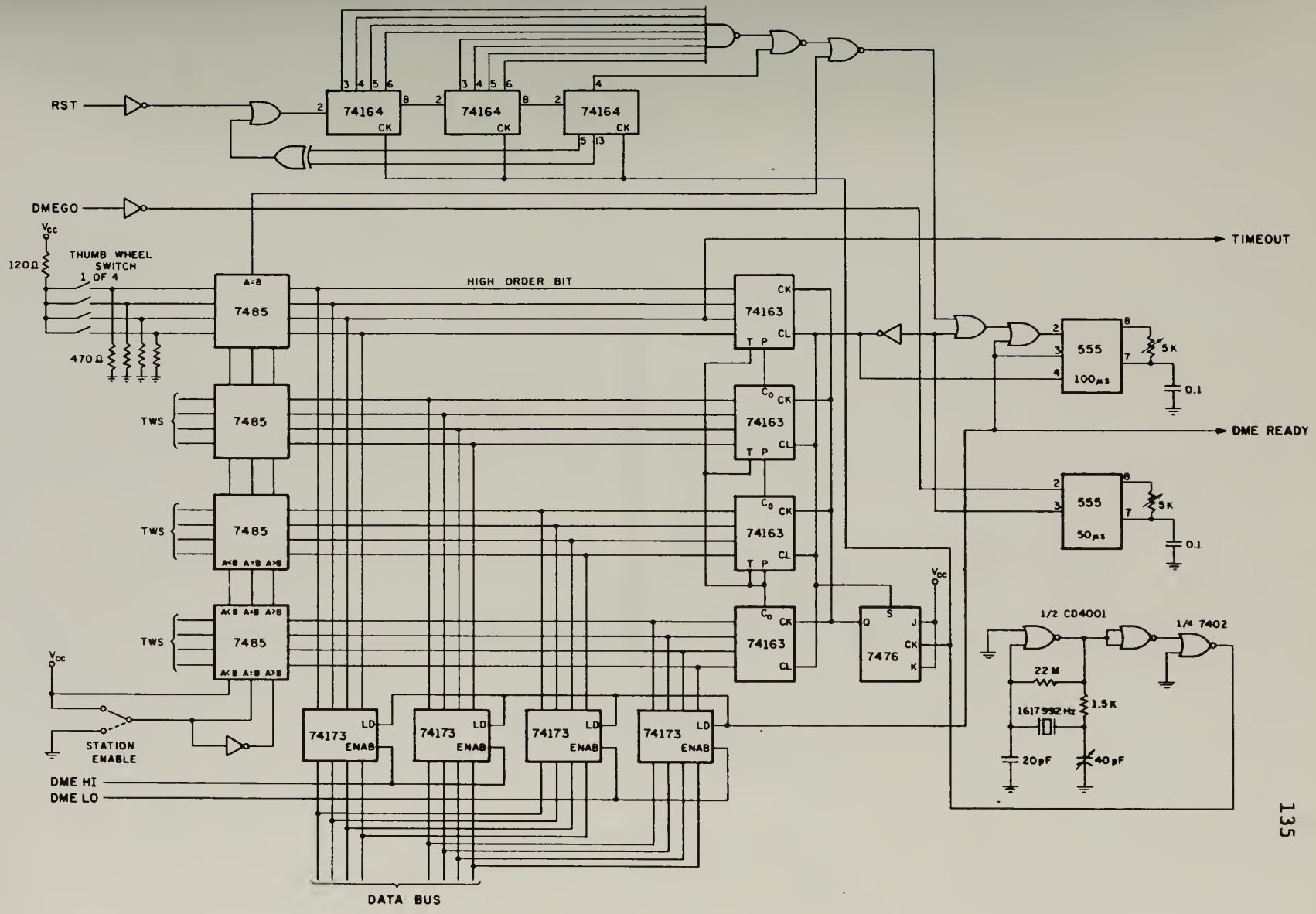
NOTE: ALL 74170 CONNECTIONS ARE IDENTICAL


FROM
REGISTER FILE
A
B
C
D
7448
a b c d e f g
V_{cc}
12 LINES
TO
CATHODE
MULTIPLEXOR


Vcc
1
2
3
8
7
6
555
1K
3300
.01
74163
CL
CK
D C B A
ENAB 1
ENAB 3
ENAB 2
RA0
RA1
15
14
13
12
N/C
11
10
9
8
7
6
5
4
3
2
1
0
74154
7404
75492
7404
75492
CATHODE
MULTIPLEXOR
FOUR DIGITS
PER LINE

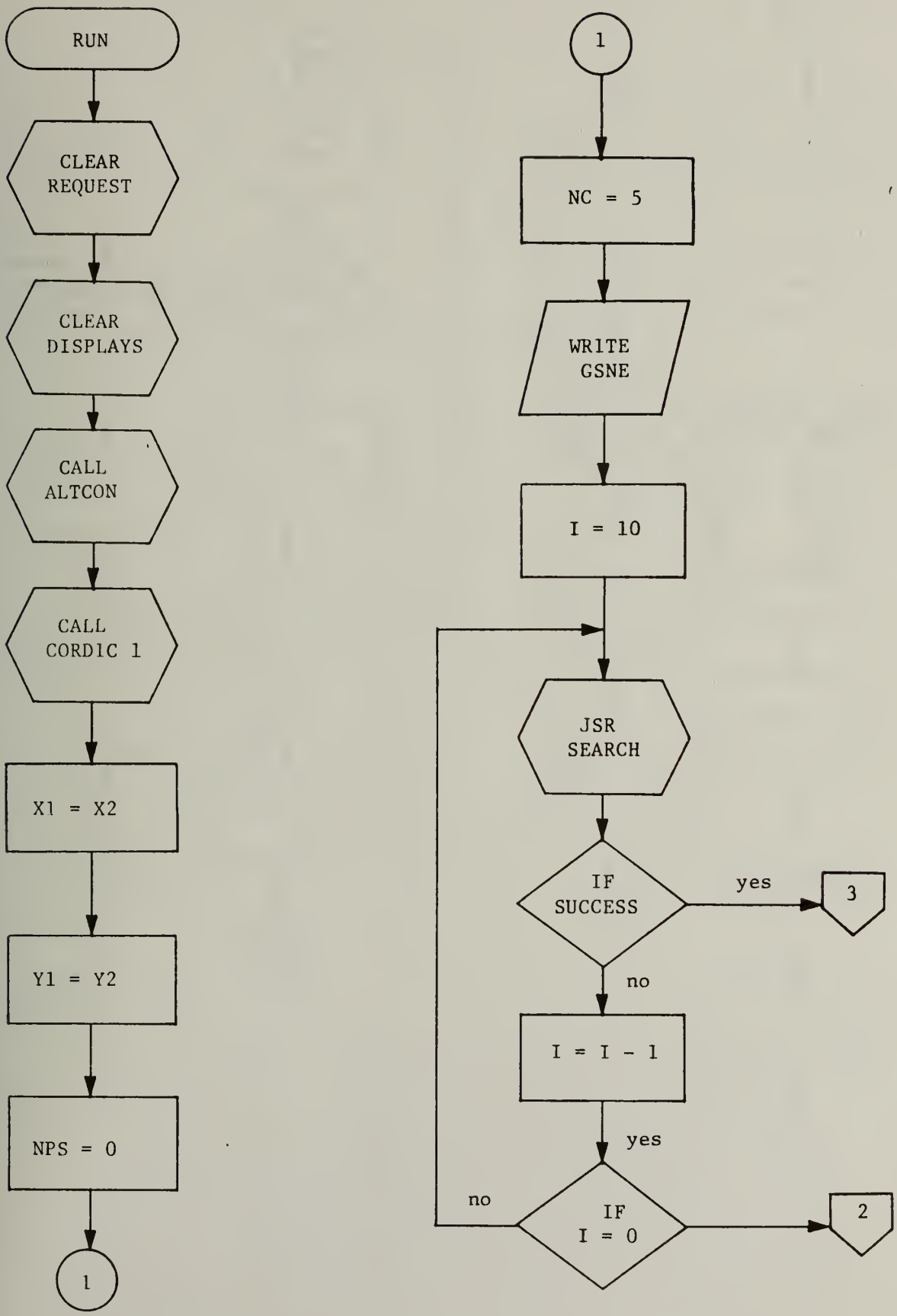
APPENDIX 5

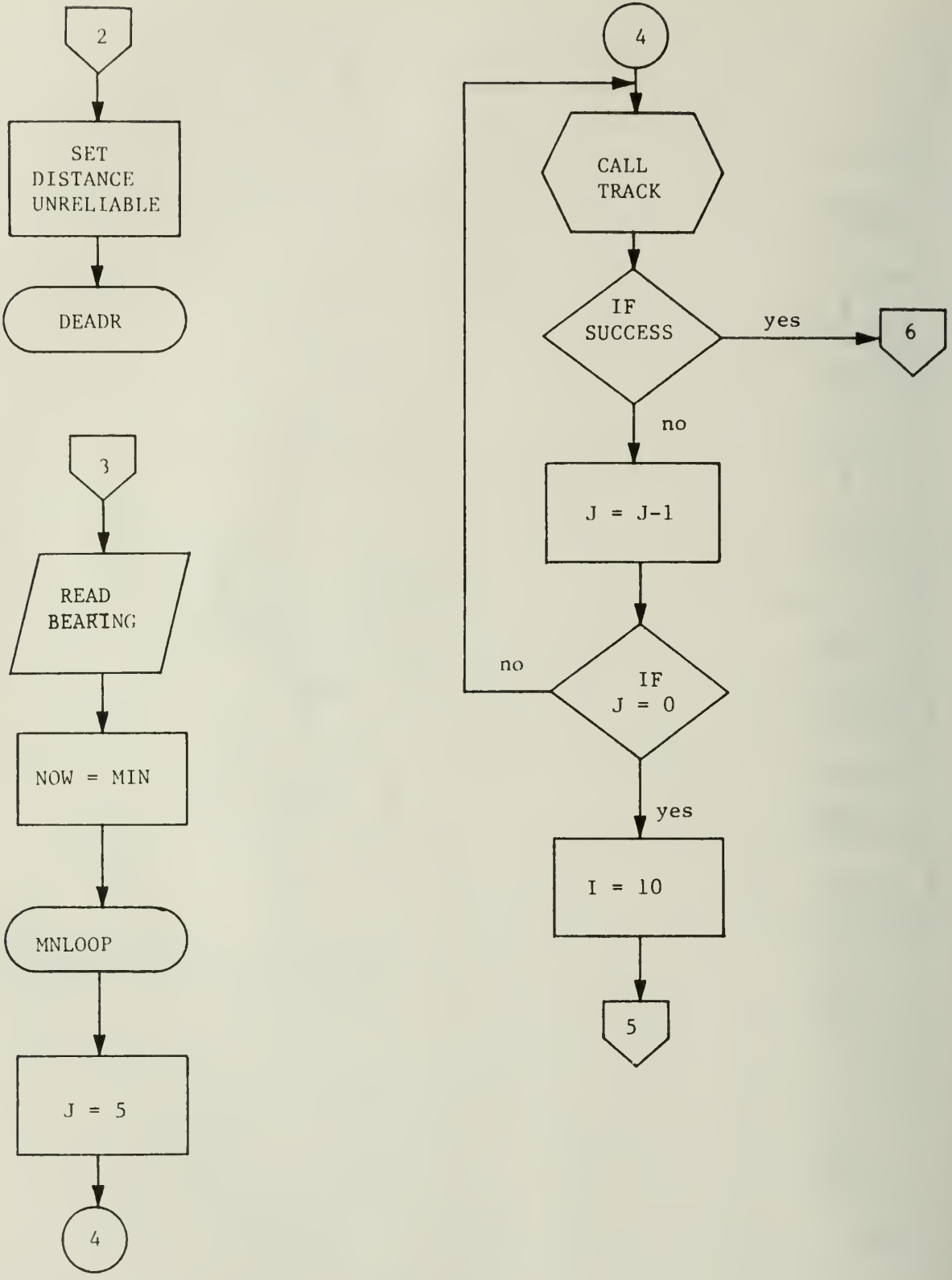
DME Simulator Circuit

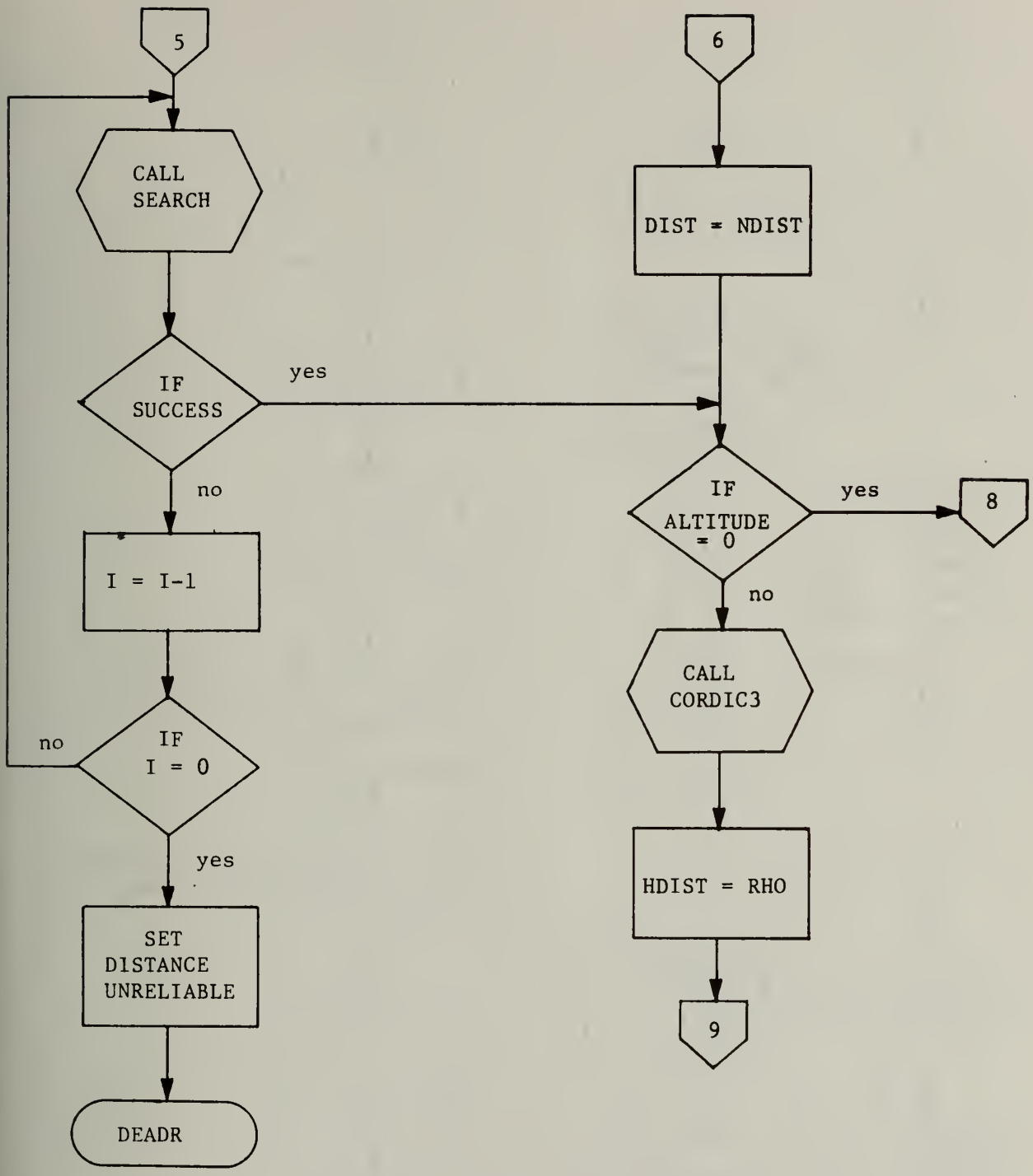

RST
74164
CK
DMEGO
Vcc
120Ω
THUMB WHEEL SWITCH
1 OF 4
470Ω
7485
A=B
A<B A=B A>B
TWS
HIGH ORDER BIT
74163
CK
CL
T P
Co
TIMEOUT
555
100μs
5K
0.1
DME READY
50μs
7476
S
J
Q
K
1/2 CD4001
1/4 7402
22M
1.5K
1617992 Hz
20pF
40pF
74173
LD
ENAB
STATION ENABLE
DME HI
DME LO
DATA BUS

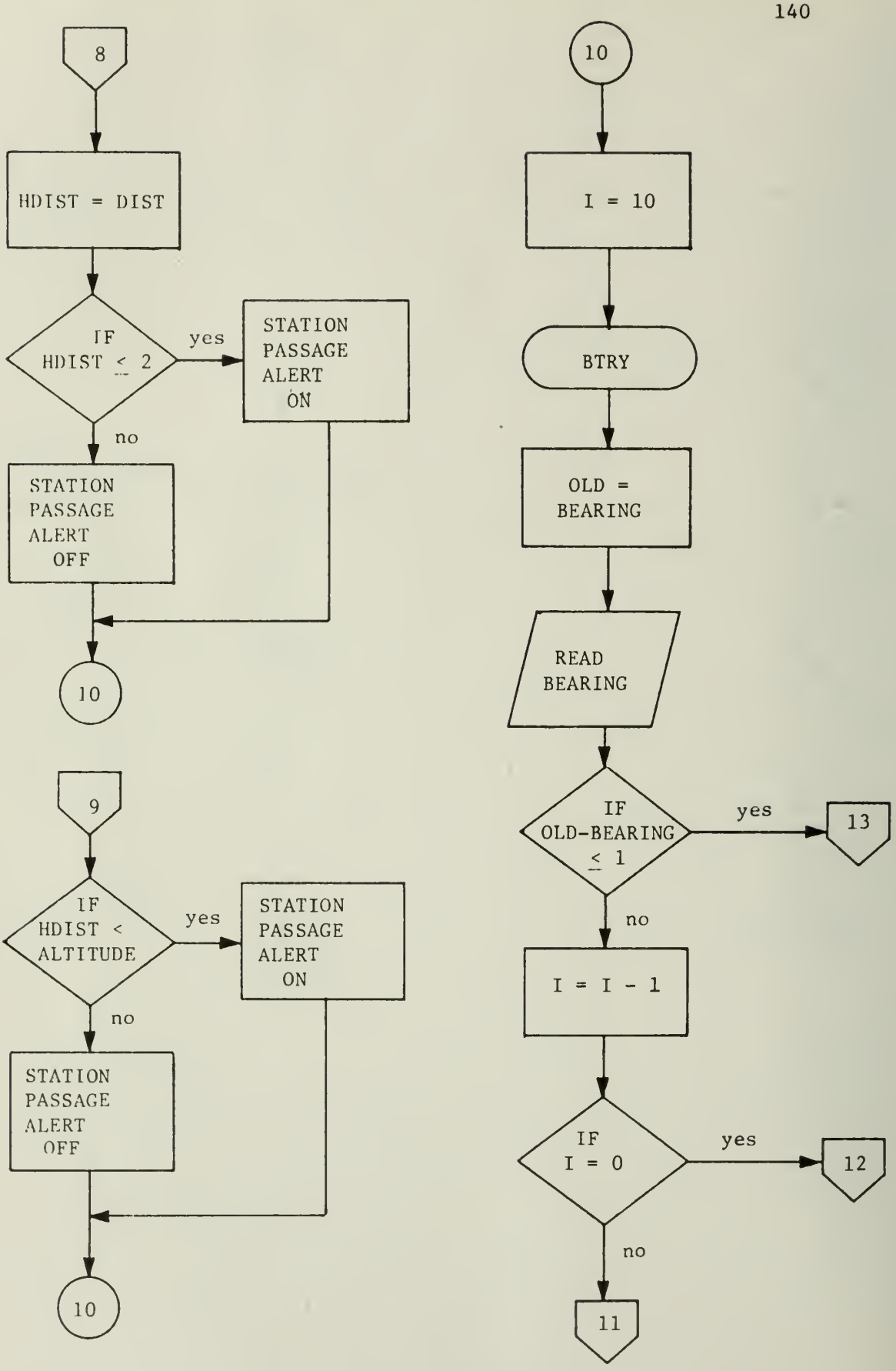
APPENDIX 6

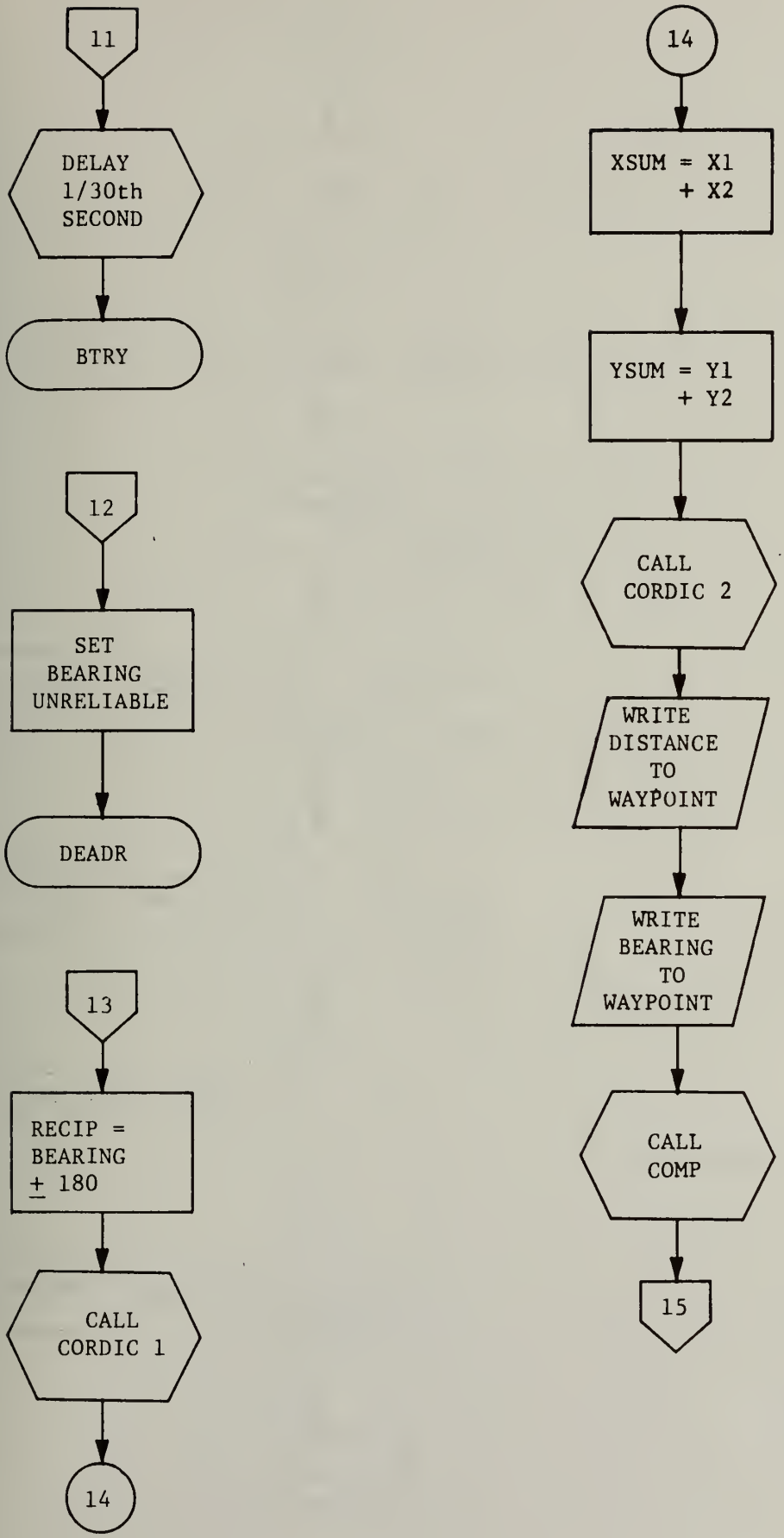
MAN System Program Flowchart

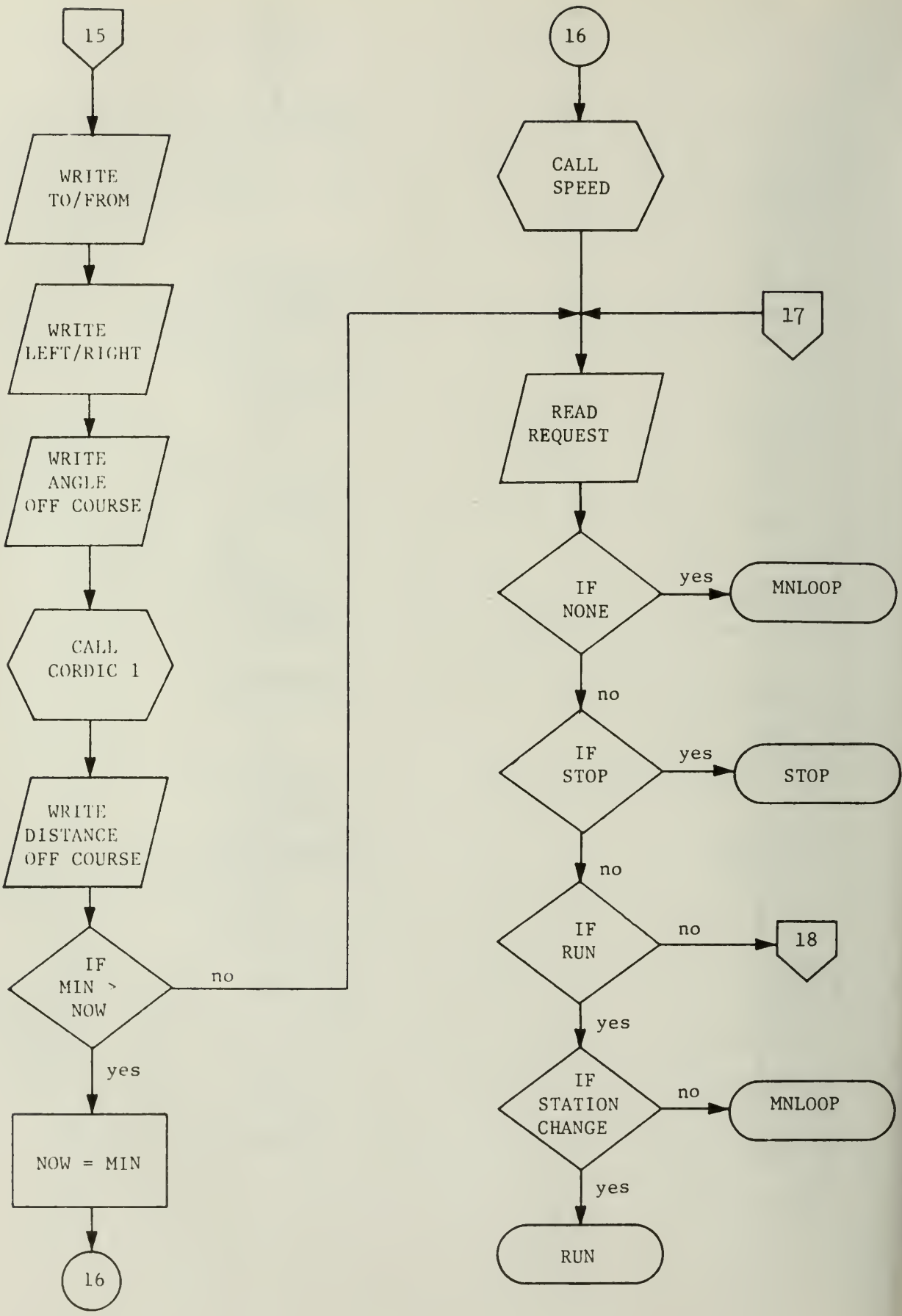

RUN
CLEAR REQUEST
CLEAR DISPLAYS
CALL ALTCON
CALL CORDIC 1
X1 = X2
Y1 = Y2
NPS = 0
1
1
NC = 5
WRITE GSNE
I = 10
JSR SEARCH
IF SUCCESS
yes
3
no
I = I - 1
yes
no
IF I = 0
2

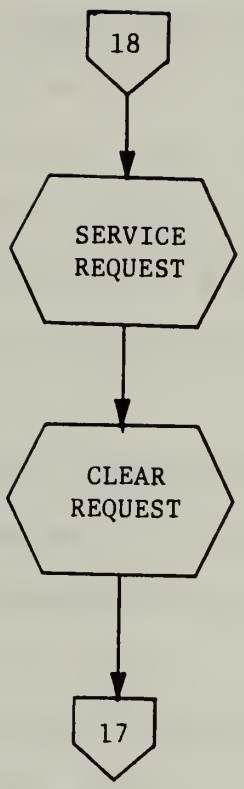

2
SET
DISTANCE
UNRELIABLE
DEADR
3
READ
BEARING
NOW = MIN
MNLOOP
J = 5
4
4
CALL
TRACK
IF
SUCCESS
yes
6
no
J = J-1
IF
J = 0
no
yes
I = 10
5


5
CALL SEARCH
IF SUCCESS
yes
no
I = I-1
IF I = 0
no
yes
SET DISTANCE UNRELIABLE
DEADR
6
DIST = NDIST
IF ALTITUDE = 0
yes
8
no
CALL CORDIC3
HDIST = RHO
9


8
HDIST = DIST
IF HDIST ≤ 2
yes
STATION PASSAGE ALERT ON
no
STATION PASSAGE ALERT OFF
10
9
IF HDIST < ALTITUDE
yes
STATION PASSAGE ALERT ON
no
STATION PASSAGE ALERT OFF
10
10
I = 10
BTRY
OLD = BEARING
READ BEARING
IF OLD-BEARING ≤ 1
yes
13
no
I = I - 1
IF I = 0
yes
12
no
11


11
DELAY 1/30th SECOND
BTRY
12
SET BEARING UNRELIABLE
DEADR
13
RECIP = BEARING ± 180
CALL CORDIC 1
14
14
XSUM = X1 + X2
YSUM = Y1 + Y2
CALL CORDIC 2
WRITE DISTANCE TO WAYPOINT
WRITE BEARING TO WAYPOINT
CALL COMP
15


15
WRITE TO/FROM
WRITE LEFT/RIGHT
WRITE ANGLE OFF COURSE
CALL CORDIC 1
WRITE DISTANCE OFF COURSE
IF MIN > NOW
no
yes
NOW = MIN
16
16
CALL SPEED
17
READ REQUEST
IF NONE
yes
MNLOOP
no
IF STOP
yes
STOP
no
IF RUN
no
18
yes
IF STATION CHANGE
no
MNLOOP
yes
RUN


18
SERVICE REQUEST
CLEAR REQUEST
17

VITA

Wayne Douglass Smith was born in Knoxville, Tennessee, on May 26, 1935. He graduated from high school in Middlesboro, Kentucky, and attended Lincoln Memorial University before joining the U.S. Air Force in 1955. He graduated from USAF Pilot Training in the Class 1956-M, and served on active duty until 1968. While on active duty, he married the former Sara Anne Cathey of Selma, Alabama. They have two daughters, Leigh Anne and Erin. While serving in the Panama Canal Zone, Mr. Smith attended language classes at the Inter-American Air Academy, and both reads and speaks Spanish.

Upon release from active duty, Mr. Smith attended Auburn University, where he received his B.S. in Mathematics in 1969. He received his M.S. in Information Science from the Georgia Institute of Technology in 1970.

Mr. Smith held the position of Assistant Professor of Mathematics and Computer Science at West Georgia College from 1970 to 1973. In 1973, Mr. Smith left this position to attend the University of Illinois. While at Illinois, he was elected to associate membership in the Society of Sigma Xi, a research society, and to membership in Chi Gamma Iota, a veterans honorary society. He also served as treasurer of the University of Illinois student chapter of the Association for Computing Machinery, and as secretary of the Computer Science Graduate Student Organization.

Mr. Smith holds a commercial pilots license, a radio amateur operators license, and is a member of Masonic Lodge No. 27, Selma, Alabama. Mr. Smith is currently employed as an Assistant Professor of Computer Science at Angelo State University, San Angelo, Texas.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-76-825	2.	3. Recipient's Accession No.

4. Title and Subtitle

AN INVESTIGATION OF THE APPLICATION OF MICROPROCESSORS TO LOW COST AREA NAVIGATION CAPABILITIES FOR GENERAL AVIATION

5. Report Date: September 1976

6.

7. Author(s)

Wayne Douglass Smith

8. Performing Organization Rept. No. UIUCDCS-R-76-825

9. Performing Organization Name and Address

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

10. Project/Task/Work Unit No.

11. Contract/Grant No.

12. Sponsoring Organization Name and Address

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

13. Type of Report & Period Covered: Ph.D. Dissertation

14.

15. Supplementary Notes

16. Abstracts

A joint FAA/Industry Task Force has recommended that area navigation become the primary method of aerial navigation in the United States by 1982. This mode of navigation has already gained wide acceptance in commercial aviation, but due primarily to high cost, has not yet become established in general aviation. This thesis investigates the possibility of utilizing an off-the-shelf microprocessor as a low cost alternative to current special design systems. The Microprocessor-based Area Navigation (MAN) System is designed around the MOS Technology 6502 microprocessor. The system software makes extensive use of the CORDIC algorithm to evaluate the trigonometric functions associated with the area navigation computations. The MAN System is constructed, programmed, and tested using simulated VOR and DME inputs in a static ground environment. Computation times on the order of 0.2 seconds are achieved, and the system operates well within the tolerances specified by the FAA. The total system cost is low enough to offer a significant economic advantage over systems that are currently available.

17. Key Words and Document Analysis.

Microprocessor, Navigational Computers, Area Navigation, R-Nav, Navigation, CORDIC Algorithms, Coordinate Rotation, Aviation, General Aviation, Waypoint.

17b. Identifiers/Open-Ended Terms

MAN System
KIM-1 System
MOS Technology 6502 Microprocessor
Peripheral Interface Adapter

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Release Unlimited	UNCLASSIFIED	149
	20. Security Class (This Page) UNCLASSIFIED	22. Price -----